A Macroeconomics of Social Contracts

A thesis submitted in fulfilment of the requirement for the degree of Doctor of Philosophy of Cardiff University

Thomas Wilkinson

September 2013

Economics Section of Cardiff Business School
Cardiff University

Declaration

This work has not previously been accepted in substance for any degree and is not concurrently submitted in candidature for any degree.

Signed ..... (candidate)

Date .....

Statement 1

This thesis is being submitted in partial fulfilment of the requirements for the degree of PhD.

Signed ..... (candidate)

Date .....

Statement 2

This thesis is the result of my own independent work/investigation, except where otherwise stated. Other sources are acknowledged by explicit references.

Signed ..... (candidate)

Date .....

Statement 3

I hereby give consent for my thesis, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

Signed ..... (candidate)

Date .....

Copyright © 2013 Thomas Wilkinson.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".

A copy of this document in various transparent and opaque machine-readable formats and related software is available at `https://drive.google.com/folderview?id=0B05vI0R35YDqWTBVT1Q1WE01d3c&usp=sharing`.

Abstract

This thesis sets out the case and foundations for a new way to think about, and model, Macroeconomics. This framework aims to describe the fluctuations and differing growths of economies, not in terms of the choice and exchange of Microeconomics, but rather in terms of the enforcement relationships that allow that exchange and other cooperation between people. It first establishes just why this is necessary, with a thorough methodological critique of the way Macroeconomics is done right now. It then presents computational models of two presumably competing kinds of enforcement relationship. The first of these is the third party supervision that we are most familiar with as *enforcement* from every day life, and which has received some of the longest running philosophical discussion. This hierarchical model reproduces economic fluctuations, through occasional collapses of large parts of the hierarchy. To assess the scientific merit of this model on the terms of conventional Macroeconomics, I develop a compatible hypothesis testing strategy. The second kind of enforcement considered is what would commonly be called *peer pressure*. For this I derive a preliminary result, that would allow further development of an overarching research program.

Acknowledgements

I would like to thank Prof. Huw Dixon, for his advice and oversight, and Prof. Patrick Minford, both for his advice and for many instructive debates.

Contents

List of Figures

List of Tables

Chapter 1

Past Failures and Future Possibilities

1.1 Introduction

This introductory chapter lays the foundations for the subsequent exploration of a new paradigm in modelling Macroeconomics. That paradigm is based not on the descriptions of choice and exchange that have occupied microeconomics, and by extension conventional macroeconomics, but on the enforcement of the agreements that make both exchange and most other social cooperation possible.

Section 1.2 explains the setting, a scientific discipline that has not yet achieved either of its intended goals. Section 1.3 sketches the ideas that underlie an alternative paradigm for this discipline. A review of models from Network Science, a promising language to use in developing this new paradigm, starts with notation and terminology in section 3.3. This is followed by descriptions of static models in section 3.3.1, and dynamic models in section 3.3.2. Section 1.6 concludes, and ties these foundations into the development of an enforcement based Macroeconomics in subsequent chapters.

1.2 Motivation

1.2.1 Perceived Failings of Anti-Recessionary Policy

In 2007 and 2008, the economic powerhouses of the OECD saw the sudden onset of the greatest recession for nearly a century. A downturn that started in the housing and construction industries was exacerbated by the 2008 failure of Lehman Brothers, an institution too big and connected to fail. Even after other financial institutions were prevented from failing, and began to control their liabilities, a further sovereign debt crisis in Europe started to threaten the world economy; the recession dragged on. There could be no more powerful reminder, than this avalanche of recessions and solvency crises, that Macroeconomics has yet to make progress on its founding question of stabilising the economy. But, 2008's reminder came after two decades of increasing optimism, some might say hubris, inspired by the relative stability of western economies over the period known as the *Great Moderation*. So ebullient was the macroeconomic orthodoxy that its figurehead and president of the American Economics Association, Robert Lucas, famously declared in his 2003 presidential address that the "central problem of depression-prevention has been solved" (Lucas, 2003).

With the onset of the recession this overconfidence was rewarded with serious reprimands from the lay public and even some prominent microeconomists, as Paul Krugman in the New York Times and James Heckman in the Observer both lambasted the New Neoclassical Synthesis school for theoretical inconsistency and disconnection from empirical evidence (Krugman, 2009; Heckman, 2010). Criticisms generally highlighted the lack of *realism* in certain modelling assumptions, but put most emphasis on the fact that the recession had not been predicted. This former criticisms generally missed the fact that those particular unrealistic assumptions were actually too dependent on others to even be compared to empirical reality. Meanwhile, the latter neglected to mention that macroeconomic models treated the drivers of recession as entirely unpredictable, making predictive failure at least consistent with these models,

even if it undermined their utility. None questioned the deeper foundations of the models or Economics generally, the concepts of price-equilibrium and aggregated supply and demand. Indeed, many mainstream macroeconomists were vocal in their defence of the *Dismal Science*. But their defences seemed predicated on *what we know* about the economy, which in the details actually consisted of *what theorists take for granted*.

The problems of 2008 were, of course, nothing new to economics. Undesirable fluctuations in the economy were first academically acknowledged in Sismondi's critique of classical political economy in the early 1800s. The popular liberalism of the time quickly answered this critique, as Dunoyer argued for cycles of *activity* and *relapse* within the classical theoretical framework. Whereas Sismondi saw fluctuations as crises that required government intervention, Dunoyer made them inevitable and irrepressible features of a healthy entrepreneurial economy (Benkemoune, 2009).

This attitude wasn't seriously challenged again until the rise of Keynesianism following the Great Depression. It was in the work of Keynes that Macroeconomics as an independent discipline came into existence, with its principal goal to prevent subsequent recessions. With the observation of the *Phillips Curve* relationship between inflation and unemployment, Keynesian Macroeconomics came to believe that it had a tool, in inflation, with which to manage unemployment. Keynesian Econometrics then began to model the economy as the conjunction of goods markets that could be influenced by government spending, and financial markets influenced by government interest rates and the monetary stock. The structure of these models was determined by theory, but the specific numbers that parametrised them were determined by statistical treatment of macroeconomic data in the spirit of the Phillips Curve. Briefly, Macroeconomics believed that it had addressed the issue of Economic fluctuation, but this came undone in the 1970s when the Phillips Curve relationship was broken by simultaneous inflation and economic stagnation.

Macroeconomics adapted, and the response to the Phillips Curve's failure focussed on questions of whether the correlations underlying Econometrics truly represented

causation. This coalesced into the famous Lucas Critique which argued for *microfounded* macroeconomic models developed from microeconomic theory. The Dynamic Stochastic General Equilibrium models that emerged from this research programme coincided with the Great Moderation, and so were at the heart of the confident Macroeconomic profession that was caught so off guard by the 2008 crisis.

Whether the crisis could have been predicted or not, the fact remains that it was no more prevented by Macroeconomics than the crises that inspired Sismondi, Keynesianism, or the Neoclassical resurgence that followed it. By definition this can be nothing but a failure on the problem of recessions, for which Macroeconomics was originally conceived.

1.2.2 Perceived Failings of Development Policy

Although the fluctuations of the "business cycle" have remained the dominant topic in modern Macroeconomics, it has another great interest that harks back to the very origins of Economics. Adam Smith's seminal treatment of the economy, in book three of his great treatise, focussed on the growth in productivity of nations over time. Lucas, as the representative neoclassicist, more recently reconfirmed the topic (Lucas, 1988). Growth is studied extensively in the context of Western and emerging economies, but is most visible and relevant in the context of Less Developed Countries where greater growth would arguably make the greatest difference to quality of life.

Nevertheless, the past fifty years of Development Economics can be characterised as a flurry of silver bullets (or perhaps buckshot), with each only seen to have missed the target once the smoke had cleared. Only in the past decade has prevailing opinion swung away from hopes of such a panacea toward an appreciation of more nuanced answers. Nevertheless, the last of these silver bullets, the so called Washington Consensus, continued to have some currency in the policies of the world's major development institutions until the financial crisis of 2008. This was despite discontent bubbling

up to lay attention in 2002, with the release of popular books by Stiglitz (2003) and Easterly (2002) — though both held quite different views on how else to proceed.

The first silver bullet of Development Economics was the injection of capital to make up for a shortfall of internal investment, as recommended by the Harrod-Domar model. This was designed for post-war Western countries with massive pools of trained labour but, in Development Economics, was used to advise policies in countries with little human capital or institutional infrastructure. African fields produced a bumper crop of rusting donated machinery. Next came the Solow model, also originally intended only for the United States, which suggested that only short term growth could be influenced by capital accumulation, and then only up to a point. It promised permanent growth only as a response to improving technologies, and counselled sitting back to let interest rates drive what growth could be driven. Poor countries remained poor. Following Lucas' growth accounting, human capital was postulated as the link Solow had missed, and so massive aid investment in education was the response. Yet, the missing productivity turned out not to be education alone (Easterly, 2002). Finally, with the apparent stabilisation of Western economies under market liberalisation and controlled inflation, the panacea became imitating this model. The result was recessions across the developing world (Stiglitz, 2003).

Stiglitz's attack on the Washington Consensus did not end with his best-seller; in 2004 he arranged for a round table discussion of Development policy, between an international team of esteemed Development academics. This meeting was embedded in the Universal Forum of Cultures in Barcelona, with the enthusiastic support of European politicians, and so the moniker "Barcelona Consensus" was suggested for its recommendations (Serra and Stiglitz, 2008). This framework emphasised that Development policy be led not by the donors, but by the individual recipients of aid, to ensure they were tailored to context. Specific obstacles to growth were to be identified case by case, and microeconomic tools used alongside macroeconomic theory to identify how best to leverage them. Finally the target in each case was to institutionalise the reform,

rather than trying to force it through simple legislation.

Despite all this vocal opposition from two former World Bank employees, it seems that the death knell for the Washington Consensus didn't come until the whole Neoliberal perspective was challenged, in the wake of the 2008 global recession. Whether rightly or not, deep public dissatisfaction was turned against Growth Economics as well as the economics of recessions. This groundswell against the Washington Consensus was finally recognised by the international establishment in the so called Seoul Consensus of 2010 (of the G20 members, 2010). The Leader's Declaration of the Seoul Summit put special emphasis on context, through countries' self-determination of Development policy:

> The Seoul Development Consensus for Shared Growth that sets out our commitment to work in partnership with other developing countries, and LICs in particular, to help them build the capacity to achieve and maximize their growth potential...

It goes on to stress structural considerations in the execution of these *partnerships*, with contextual prioritisation:

> ...in particular: strengthening bilateral and multilateral work on surveillance covering financial stability, macroeconomic, structural and exchange rate policies, with increased focus on systemic issues...

Despite this turn toward planning, it retains an emphasis on a regulated private sector's preeminent role in economic convergence, leaving it more balanced than the Washington Consensus's free markets first doctrine.

Assuming that the earlier policy recommendations actually represented the conditions that worked for the development of the western economies, what was so different about developing countries that the same policies should have failed? The answer to this

question might be quite obvious to subjectivists in Anthropology (Fforde, 2005), but anyone hoping for generalisable mechanisms of macroscopic society might well hesitate to open the pandora's box of other potentially relevant variables. Robinson and Acemoglu (2012) provide a summary of some common choices — geography, culture, and as yet unknown market failures — but come to the conclusion that institutions are the favourite cause for the divergence of poor and rich countries[1]. They suggest that tensions between *de jure* and *de facto* power can lead to significant inefficiency.

Since the Wrong (1967) critique, Sociology has been willing enough to look for mechanical explanations of macroscopic social phenomena that economics refuses to consider. Network Sociologists might not be as hasty as Robinson and Acemoglu to draw a line between culture and institutions, with informal social institutions potentially interacting with formal ones. Indeed, Putnam et al. (1994) suggests that failed convergence can be caused by an inherent lack of *social capital* in a particular culture, causing frictions in the operation of public institutions. Whether formal or informal institutions are the weakest link, as the orthodoxy in Economics look for new direction, post recession, both Robinson and Acemoglu's and Putnam's ideas look like fertile ground, so long as the right tools are available to satisfy the Seoul Consensus' call for contextual specificity. I shall return to the idea of social capital in section 1.3.

1.2.3 Can Economics do better?

Ultimately a scientific research project must be based on some methodological assumptions, as these determine what evidence is relevant in support of any argument —including the method itself. It is now generally agreed that *foundationalist* schemes like Popper's falsificationism, which try to give a fundamental prescription for the scientific process, are too limited in scope. This prevents us from rejecting any research project for its methodological foundation, only for inconsistency with any such foundation

[1] Survey evidence certainly suggests a correlation between institutional trust and wellbeing (Hudson, 2006)

(Caldwell, 2003; Dow, 2004). Nevertheless, scientific discourse loses its purpose if anything goes, with different research projects using different language and following different agendas (Dow, 2004). The failures described above suggest that something may be quite wrong with the way Economics is being done. So, in chapter 2 of this thesis I frame key theoretical findings in a methodological argument which confirms this suspicion. Rather than leave Economics up in the air, and despite the relativism of the methodological community, I also make the case for a fundamental scientific prescription that would give structure to change.

Chapter 2 suggests that many of the drawbacks of the prevailing Dynamic Stochastic General Equilibrium approach to modelling arise from the assumption of homogeneous agents connected only indirectly via the omniscient Walrasian auctioneer. Where do we end up when we depart these assumptions? First of all, without equilibrium prices simply imposed on the system from the top, we find ourselves looking to the forces between agents for stability. What's more, we lose the convenient fallacy of composition that lent the macroeconomy exactly the same behaviour as individual agents; so the regularity we observe in real economies need not come from a strict optimisation problem imposed from the top. Instead, it is now possible to turn to the idea of *emergence*, from Complexity Science, as a mechanism by which mostly selfish local action leads in the aggregate to the global stability of Smith's *invisible hand*.

The canonical example of believable *emergence* comes from *Self Organised Criticality* in Physics, itself exemplified by the Sandpile model of Bak, Tan and Wiesenfeld (Bak et al., 1993). In such non-equilibrium systems a *critical point* state, where components can be correlated across the whole system, is an attractor; so even though grains of sand falling on a peaked pile are likely to cause cascades of sand spreading across large areas of the surface, their subsequent accumulation returns the pile to the same unstable shape. Hence, in the aggregate we see the emergence of regular, albeit unstable, behaviour that is not obvious from the properties of its components. Without assuming the global synchronisation of prices, it is mechanisms like this that must be

trusted to return the economy, and society to the status quo after financial, political, or cultural *cascades* (Kirman, 2010; Gatti et al., 2011). The problems with conventional Macroeconomics make complex systems models an appealing alternative.

A central feature of complex systems is interrelationships between the constituent parts of the system that mean their behaviour cannot be simply aggregated. In order to precisely describe these systems of relationships, models make use of the mathematics of *Graph Theory*, which gives a clear language to describe networks of relations. Economics has recently begun to see an explosion of interest in *Network Theory*, which is sometimes classed as a sub-discipline, but includes non-dynamic models. From section 3.3 onwards I will give an overview of Network Science as applied to Economics, to provide support and contrast to the more specific research program that I will develop.

1.3 Incentives, Institutions and Enforcement

In the remainder of this chapter I will outline several different perspectives on how interaction between heterogeneous individuals can impact on the economy. I, however, take inspiration for my focus from Easterly; when we strip away the invisible hand *as an axiom*, economics' spirit might well be captured by the maxim "people respond to incentives". If people respond to incentives, then which incentives align behaviour which is collectively desirable with that which is individually desirable when we no longer assume these to be the same? The obvious answer is that where it prevails, the collective good is enforced by some mechanism that reduces the incentive to the individually preferred action. These enforcement mechanisms may be greatly varied, and it would be hard to deny that moral and religious beliefs play an important role too nuanced to model. Nevertheless, if we subscribe to an evolutionary model of enforcement institutions then it is hard to believe that moral values can persist against immoral mutant beliefs unless they are backed up by some more tangible mechanism supporting them. I propose that we can describe these more tangible mechanisms in terms

of just two general descriptions: third party enforcement, where agreements between individuals are enforced by the threat of punitive action from a third party with asymmetric power to the individuals; and peer-pressure based enforcement, administered by a group against its members who stray from some prescribed behaviour. In both cases asymmetric power is needed (either inherent, or in numbers) in order to prevent the disincentive itself being disincentivised by retaliation.

A formal description of peer-pressure based enforcement emerges from the relatively young, but extremely popular topic of Social Capital, in Sociology. In the definition of Social Capital by Coleman (1988), the collective good is described in terms of obligations to reciprocity:

> If A does something for B and trusts B to reciprocate in the future, this establishes an expectation in A and an obligation on the part of B. This obligation can be conceived as a credit slip held by A for performance by B. If A holds a large number of these credit slips, for a number of persons with whom A has relations, then the analogy to financial capital is direct. These credit slips constitute a large body of credit that A can call in if necessary —unless, of course, the placement of trust has been unwise, and these are bad debts that will not be repaid...

Of course a purposive agent must not only believe that their counterpart recognises an obligation but must also believe that they have incentive to abide by it. This idea was formalised in the work of Robert Axelrod and Anatol Rapoport, which pioneered Evolutionary Game Theory: various computer programs (representing particular strategies), interacted in a strategic situation where ongoing cooperation was threatened by instantaneous rewards to breaking that cooperation, and those that were most successful were selected. The most successful of all was the Tit-for-Tat strategy, which consisted of cooperating until the opposing program broke cooperation, and then always breaking cooperation. Although there was no forward looking rationality to the strategy, through

evolution this strategy had come to effectively collateralise instantaneous cooperation by the future value of the relationship. This collateralisation can be extended to groups of others, as Coleman does with his concept of *closure*, whether the parties to a relationship both have relationships with a common third party; because whether a partner abides by obligations is an observable behaviour, its violation can be punished by any observers who also see it as an obligation. In Sociology this punishment is generally assumed to be ostracism by sufficient numbers of the agents' community to disincentivise not meeting the obligation. Compliance of observers to this sanction is then itself enforced by a second order obligation to conformity, according to which failing to sanction incurs a sanction too. In this way, pro-social behaviour is collateralised by the future value of many more relationships than just the ones affected by the specific action of the agents. This same idea was captured quite concisely by Cicero two millennia before Coleman: "There is no duty more indispensable than that of returning a kindness. All men distrust one forgetful of a benefit" (Putnam et al., 1994).

In contrast to peer-pressure's enforcement between equals stands the sovereign of Thomas Hobbes (1651), a third party capable of meting out punishment without danger of retaliation. Hobbes argued that this sovereign would ensure the rule of law out of pride for the prosperity of their nation. But, John Locke (1821) later argued that it was the threat of having their third party enforcer status removed that was motivation, and that this represented a *social contract* between sovereign and subject.

Ken Binmore's epic work *Game Theory and the Social Contract* redefines the *social contract* of Hobbes and Locke as a particular equilibrium selection convention of the massive strategic-coordination problem inherent in human society, the so called Game of Life (Binmore, 1994, 1998). He therefore includes third party enforcement, peer-pressure, and any other enforcing belief under this term, which is why I have taken it for the title of this thesis.

The elegance of peer-pressure as an enforcement strategy should now be more apparent, in that it harnesses collective action for a greater good, with no greater punitive

threat than ostracism from collective-controlled resources; all without relying on the investment of power in a particular individual who is thereby enabled to extract rent. That being said, in reality we recognise that groups on all scales of society also choose leaders to be third party enforcers, and may well cede to them power beyond any inherent in their situation[2]. The presence of leaders, and leaders' leaders (etcetera) give societies a hierarchical structure. For power to be ceded freely in a world of even bounded or evolutionary rationality, there must be some gain in efficiency from so doing. Binmore (1998) briefly mentions the usefulness of an executive, but generally considers social hierarchy a remnant of our feudal past, and maladapted to modern society. In this he perhaps overestimates the ability of a leader with only *de jure* power to smooth the volatility of the *de facto* mob[3], and that the present power of our executives might well be a Pareto improvement over a freer society. Focussed on individual institutions, rather than society as a whole, this idea is reminiscent of the New Institutional School of Economics, which treats firms not as atomic entities, but as coordination mechanisms (Ménard and Shirley, 2005).

As both enforcement mechanisms ultimately achieve the same purpose, I propose that the local balance between the formal and informal economies, and the efficacy of formal institutions, may be determined by the local competition between formal and informal enforcement mechanisms. This competition could inform models of institutional effects on economic activity and provide a description of the institutional differences between countries. That is, it promises an approach to the second great question of Macroeconomics, the differing growth rates of different nations. Meanwhile, practical models may be developed that might explicitly suggest an optimal

[2]To clarify this point, it is worth imagining a world stripped of the social norms and laws we take for granted. We might note that here no possession exists that can imbue power without simply being vulnerable to theft by a stronger group of other. For example, the influence that comes with great wealth only exists when that wealth cannot simply be taken by a temporary alliance with greater combined strength. Because no person is strong enough alone to capture and keep the wealth of a nation, we might imagine that every dictator rules with the consent of a majority of strength —or perhaps through inertia from a time when they did.

[3]Or, indeed, its capacity to temporarily adopt other maladapted strategies in a dynamic setting —an obvious extension of his argument.

balance of formal and informal stabilisation mechanisms, minimising the day to day risk that people face. It is this idea that inspires the research program developed in this thesis, but in chapters 3 and 4 I find that third party enforcement alone might be enough to produce the fluctuations that define the other great question of Macroeconomics. I then begin to work on describing peer-pressure based enforcement in chapter 5, with the hope of setting both modes of enforcement competing in a growth model.

The remainder of this chapter puts this proposed macroeconomic paradigm in more intuitive terms. It does this by comparing the *stories* told by my enforcement based macroeconomics, and by more traditional conceptions.

1.4 Answers to the question of Recessions

I have suggested that the great questions of macroeconomics remain unsatisfactorily answered. I have pointed out an aspect of human interaction that might lead to new answers. But, getting the reader thinking along these new lines will need a more tangible illustration of both points. In this section, I will be much more specific about how both traditional macroeconomics and my proposed paradigm go about answering the question of our time; why was the world hit by recession in 2008?

1.4.1 The Keynesian Story

The original Keynesian school suggested several causes for recession. But, the most intuitive of these, and the one most used in explanation of the recent crisis, is that of "Animal Spirits" —a call to the dependence of human behaviour on instincts and heuristics. According to this view, people's investment is driven by their confidence in the economy. So, a Keynesian might argue that the crisis of 2008 corresponded to a dramatic change in the, mostly arbitrary, confidence of consumers in the economy's growth. That is, some investors suddenly got spooked that property prices might not

continue to rise, and this belief quickly spread throughout the economy, thereby undermining the value of the sub-prime assets that the financial sector had become dependent on.

It is, of course, hard to argue that participants in the economy behave without a full knowledge of that economy or its current state, and that instinct and heuristic would have to take the place of rational decision making. But, the Keynesian story doesn't fall down, it falls short: it is easy to talk about collapsed confidence, but a formal model is not so easy. A widely accepted model of how individuals' confidence is determined has not been forthcoming. Nor have models of how confidence might spread between individuals received much attention in Economics.

Outside of economics, contagion has been more rigorously studied, and these models could presumably be extended to the spread of confidence through an economy. But, these models are unified in Prakash et al. (2012), giving the conditions for wide propagation across an arbitrarily connected society and arbitrary transmission mechanism. Unfortunately for the Keynesian story of consumer confidence, these models show one of two behaviours divided by an abrupt transition: either the transmission is relatively contained, or it becomes an epidemic across the whole society. When it comes to collapses of economic confidence, we see a whole range of scales between local and global —a scale free property that will be elaborated on in chapter 3. This certainly suggests that it is not in contagion versus non-contagion of falling confidence that drive recessions.

The problem of crises of confidence, is precisely that they are not well defined. But, the model I will outline in this thesis could be interpreted as a model of recessions driven by lost confidence, and so not wholly incompatible with the Keynesian story. The difference is that the confidence (or lack thereof) is quite well justified, and it is the reason for its loss, rather than the confidence itself which is modelled.

1.4.2 The Neoclassical Story

In Neoclassical Economics, the economy functions perfectly so that prices reflect the valuations of all consumers and producers. This view of the world is obviously hard to reconcile with dramatic changes in the economy. Rather than being driven by the behaviour of any agents within the economy, these changes have to be driven by outside, *exogenous*, forces. What might these forces be, is never entirely clear, particularly when the disruption to the economy is as great as that in 2008.

Other than throwing our hands in the air and pleading ignorance, as many leading Neoclassical economists did during the crisis, the best Neoclassical story might be to label the *exogenous* disruption financial. That is, it was a technological change in the financial sector that suddenly reduced the economy's productivity. This might be interpreted as a sudden realisation that sub-prime lending was dependent on bad maths —a by no means uncommon narrative. But, arguably the maths of sub-prime lending had been known from the start. It's problem was that the assets' values depended on diversification by holding many risky assets that were only related through the overall economy. So although sub-prime assets could have been an accelerant of the crisis, they lost value because of a change to economic circumstances rather than a change to our understanding of them. Like other Neoclassical invocations of *external* disturbances of the economy, like oil prices or misjudged monetary policy, it is apparent that the disturbance was anything but external.

1.4.3 The Enforcement Story

My story is simple enough. While the banks we regard as responsible for the 2008 crisis may be treated as distinct entities by the law, they are in fact highly interrelated —in ownership, in interaction, and in intercommunication. Not only can we see individual banks as hierarchical structures of executives enforcing agreements between managers, but I would argue that the many informal interactions that must take place

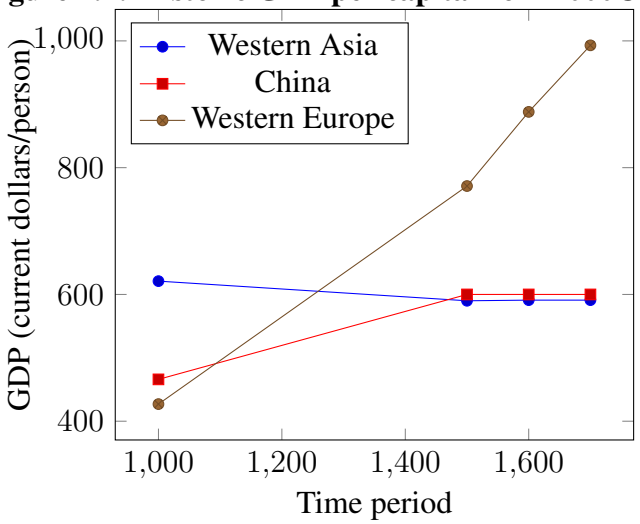
between employees of these banks can only be sustained if there are informal hierarchies spanning the banks —more on this in chapter 3. With this highly interconnected hierarchy spanning the whole economy, if one senior individual is rendered unable to enforce agreements between their subordinates then it could have wide-ranging repercussions. I argue, and my formal model shows, that the loss of enforcement of productive agreements at these senior levels can cause an avalanche of unenforced agreements throughout the hierarchy.

The 2008 crisis started, in my view, with some small issue affecting a very senior executive or shareholder in the financial sector. This individual was then no longer able to enforce agreements between those looking for them to lay down the law. Without these agreements, those subordinates no longer had the means to give them asymmetric power to individuals below them in the hierarchy. Without this power, the subordinates could no longer act as enforcers for the agreements between their own subordinates, and so the avalanche continued. In the end, much of the informal collaboration and sharing of information that had made the financial institutions function could no longer happen. This led to a fall in transactions over property that was superficially seen as an economy-wide collapse in property prices, and the subsequent recession. This collapse of trust within and between institutions is borne out by the collapse of interbank lending for a long period following the crisis.

1.5 Answers to the question of divergent growth

The other question posed by macroeconomics, is the different degrees of development shown by different countries. This is perhaps exemplified by the rapid growth of Europe during the latter half of the 2nd millennium, suddenly outstripping the formerly larger economies of China and the Muslim Caliphate(s). In this section I will illustrate both orthodox Economic explanations of this divergence, and my own. This divergence is shown in figure 1.1.

Figure 1.1: Historic GDP per capita from 1000CE



1.5.1 The Solow-Swan Story

The Solow-Swan model mentioned in section 1.2 is the mainstay of orthodox Macroeconomics's *Exogenenous Growth Theory*. It makes the stark prediction that economies' rates of per capita growth will depend only on the rate of technological change, in the long run - because capital will flow across borders until capital intensity is at an optimum level.

There is, of course, no explanation in this result for why countries might have different long run growth rates. In order to reconcile it with reality, human capital can be added to the model. This would then suggest that the reason Europe accelerated beyond the Middle East and China, was because of education levels. But, the Renaissance in Europe, that coincided with the accelerated growth, happened largely on the back of knowledge recovered from the Islamic world. There seems no compelling reason to believe that European populations should have had higher standards of education than their Muslim or Chinese counterparts.

1.5.2 The Endogenous Growth Story

Dissatisfaction with the exogenous nature of technological development in Solow-Swan and its derivatives led to the development of models in which productivity growth was identified with factors in which governments could intervene —such as human capital. In these models, the returns to investment do not ultimately die out, and so growth becomes a product of behaviour inside the model.

Endogenous growth models are diverse, but an obvious implication of a simple model with constant returns to capital would be to try and lift a country out of poverty with investment. But, as section 1.2 made clear, international development has seen massive external investment in physical and human capital that ultimately did not drive convergence. Indeed, when the European powers carved up the former Muslim world, during the colonial era, they failed to make the region converge. Similarly, the more recent failure of the Washington Consensus also undermines any interpretation of endogenous growth models as a call for market liberalisation to make less developed economies more like the models.

1.5.3 The Enforcement Story

Whereas the story of enforcement made recessions a necessary product of hierarchies, I argue that divergent growth is caused by their absence. Their absence, I propose, can be due to the presence of an alternative enforcement mechanism in the form of peer-pressure. Peer pressure might be highly effective for less developed economies, where the enforced agreements are generally within networks of peers. But, reliance on this system can lock economies in to it, and prevent them from investing in the hierarchical institutions that allow agreements over long distances, between anonymous parties.

In the second millennium case of the divergence of Europe from the Muslim and Chinese worlds, an enforcement story needs no more than the differences in population density at the time to generate a plausible divergence —an argument borrowed from

Kumar and Matsusaka (2009). At the time China and the Muslim Caliphate had vast economies, but economies based predominantly on local trade between known parties. Because populations were denser than Europe, these economies required more investment in the relationships that made up the peer pressure networks supporting trade. Europe on the other hand, with a lower population density and more long-distance exchange, would have had to invest more in formal legal systems based on hierarchical enforcement. As global technologies changed, and brought greater advantage to trade over large distances, Europe's legal hierarchy based enforcement allowed expansion. Meanwhile, the peer-pressure based enforcement of the other economies continued to demand investment to maintain essential local agreements, and thereby prevented the development of legal institutions to support long distance trade.

1.6 How to Proceed?

The first half of this chapter briefly outlined the failures of Macroeconomics and then made some broad suggestions for an alternative avenue to explore. Indeed, this exploration will constitute the remainder of this thesis and will beg continuation as an ongoing research program. Chapter 2 establishes, in firm methodological terms, why the current macroeconomic paradigm should not see the investment of resources that it does. It concludes that pluralism is what Macroeconomics needs right now. Chapter 3 then describes a model of the hierarchical third party enforcement discussed in section 1.3, that aims to address the failure of conventional macroeconomics in its first project of economic stability. Chapter 4 develops the empirical methods needed to assess the empirical value of this new approach on the terms of conventional Macroeconomics. Finally, chapter 5 begins to explore questions fundamental to developing a model of the peer-pressure enforcement also discussed in section 1.3.

Before the formal exposition of the following chapters, the second half of this chapter tried to make more intuitive this proposed approach to macroeconomics. It did this with

comparisons of informal *stories* about the recent recession and the growing divergence of more and less developed economies. These comparisons should have convinced the reader that thinking about short and long term societal change purely in terms of the goods-exchange of traditional economics is unnecessarily narrow. Instead, a simple and believable account of much more general human behaviours can quickly suggest the features we see on a societal scale. Of course, though these intuitive accounts may be convincing, we would expect more rigour from a modern scientific research program. Because the question this thesis addresses is so ambitious, and large, the following chapters take that technical rigour to very different dimensions of scientific enquiry: methodology and philosophy, modelling, and empirical testing. Such breadth and depth might at times be daunting to the reader. But, throughout, I hope referring back to this chapter's *stories* will keep the context and motivations clear.

Bibliography

Bak, P., Chen, K., Scheinkman, J., and Woodford, M. Aggregate fluctuations from independent sectoral shocks: self-organized criticality in a model of production and inventory dynamics. *Ricerche Economiche*, 47(1):3–30, March 1993. URL http://ideas.repec.org/a/eee/riceco/v47y1993i1p3-30.html.

Benkemoune, R. Charles dunoyer and the emergence of the idea of an economic cycle. *History of Political Economy*, 41(2):271 – 295, 2009. ISSN 00182702. URL http://search.ebscohost.com/login.aspx?direct=true&db=buh&AN=38809757&site=ehost-live.

Binmore, K. *Game theory and the social contract: Playing fair*. The MIT Press, 1994.

Binmore, K. *Game theory & the social contract*. MIT Press, 1998.

Caldwell, B. *Beyond Positivism*. Taylor and Francis, 2003. ISBN 9780203565520.

Coleman, J. S. Social capital in the creation of human capital. *American journal of sociology*, pages S95–S120, 1988.

Dow, S. Structured pluralism. *Journal of Economic Methodology*, 11(3):275–290, 2004.

Easterly, W. *The elusive quest for growth: economists' adventures and misadventures in the tropics*. MIT Press, 2002. ISBN 9780262550420.

Fforde, A. Persuasion: Reflections on economics, data, and the 'homogeneity assumption'. *Journal of Economic Methodology*, 12(1):63–91, March 2005.

Gatti, D., Delli, G., Desiderio, S., Gaffeo, E., Cirillo, P., and Gallegati, M. *Macroeconomics from the Bottom-up*. Springer, 2011. ISBN 9788847019706.

Heckman, J. Comment on uk government spending cuts. *The Observer 24/10/10*, page 29, October 2010.

Hobbes, T. *Leviathan Or The Matter, Form, and Power of a Common-wealth, Ecclesiastical and Civil*. Andrew Crooke, 1651. URL `http://books.google.co.uk/books?id=L3FgBpvIWRkC`.

Hudson, J. Institutional trust and subjective well-being across the eu. *Kyklos*, 59(1): 43–62, 2006. ISSN 1467-6435.

Kirman, A. *Complex Economics: Individual and Collective Rationality*. Taylor and Francis, 2010. ISBN 9780415568555.

Krugman, P. How did economists get it so wrong? *The New York Times 02/09/09*, September 2009.

Kumar, K. B. and Matsusaka, J. G. From families to formal contracts: An approach to development. *Journal of Development Economics*, 90(1):106–119, September 2009. URL `http://ideas.repec.org/a/eee/deveco/v90y2009i1p106-119.html`.

Locke, J. *Two Treatises on Government*. R. Butler, 1821. URL `http://books.google.co.uk/books?id=AM9qFIrSa7YC`.

Lucas, R. E. On the mechanics of economic development. *Journal of Monetary Economics*, 22(1):3–42, July 1988. URL `http://ideas.repec.org/a/eee/moneco/v22y1988i1p3-42.html`.

Lucas, R. E. Macroeconomic priorities. *American Economic Review*, 93(1):1–14, March 2003. URL `http://ideas.repec.org/a/aea/aecrev/v93y2003i1p1-14.html`.

Ménard, C. and Shirley, M. *Handbook of new institutional economics*. Springer, 2005. ISBN 9781402026874.

of the G20 members, L. The g20 Seoul summit leaders' declaration. Technical report, G20, Nov. 2010.

Prakash, B. A., Chakrabarti, D., Valler, N. C., Faloutsos, M., and Faloutsos, C. Threshold conditions for arbitrary cascade models on arbitrary networks. *Knowledge and information systems*, 33(3):549–575, 2012.

Putnam, R. D., Leonardi, R., and Nanetti, R. Y. *Making democracy work: Civic traditions in modern Italy*. Princeton university press, 1994.

Robinson, J. and Acemoglu, D. *Why Nations Fail*. Profile Books Limited, 2012. ISBN 9781846684296.

Serra, N. and Stiglitz, J. *The Washington Consensus reconsidered: towards a new global governance*. Oxford University Press, 2008. ISBN 9780199534081.

Stiglitz, J. *Globalization and Its Discontents*. W.W. Norton, 2003. ISBN 9780393324396.

Wrong, D. *The oversocialized conception of man in modern sociology*. Bobbs-Merrill, 1967.

Chapter 2

The Methodological Call for New Microfoundations

2.1 Introduction

As the preceding chapter explained, conventional Macroeconomics is widely seen as having not yet succeeded in either of its two great projects: mitigating the negative impacts of recessions, and encouraging economic growth in Less Developed Countries. In this chapter I carefully explain why conventional microfounded Macroeconomics, embodied in the Dynamic Stochastic General Equilibrium model, should not be expected to address these or any other empirical problems.

There are other critiques of the neoclassical microfoundations (Ackerman, 2001; Rizvi, 1997), but they tend to stop after making the points that I do in section 2.3, establishing that General Equilibrium is not realistic. I have found in debates with more conventional macroeconomists that this is not nearly enough, and that decisive criticism has to extend to the actual performance of DSGE models. The greatest contribution of this chapter, to the methodological literature, is to combine these criticisms into a single structured argument, by identifying them as addressing the distinct perspectives on describing phenomena of Holism and Reductionism —or Methodological Individualism as it manifests in the Social Sciences. This means that they can be handled separately. I address the second perspective in 2.4, and this then leaves no empirical defence for

DSGE.

I also make a further contribution, to deep economic methodology. Modern Philosophy of Science is relativistic, and reluctant to be prescriptive about the foundations of the Scientific Method. Ultimately this could allow a proponent of the Neoclassical School to argue a case for DSGE based on the prophetic intuition of the school's *big names* taking priority over other considerations. This argument defies common sensibilities but, to make this chapter a complete refutation, I undertake the much more ambitious project of giving a prescription for the Scientific Method that specifically prohibits such an argument. This prescription, in section 2.6, is essentially a refined inductive inference with emphasis on parsimony, and addresses the classic criticisms of induction. It works on both the level of subjective experience and the level of science as a social endeavour. Above all, it aims to be operational by tying into the Minimum Description Length principle for statistics, which differs from and is more sound than Frequentist and Bayesian statistics, but retains many useful techniques from each. To my knowledge, this is the first piece in the Economics literature introducing MDL.

2.2 Why explicit methodology might matter

Economists are trained to think of institutions evolved to an optimal state, and to prefer distributed institutions over centralised ones. It could be seen as natural, then, that many should think of methodologists planning the institutions of Science as either redundant or inefficient. The evolutionary arguments are of course over-simplistic, because they assume that the environmental pressures guiding the evolution of scientific methods align perfectly with what society needs from Science. In this section I will explore some of the reasons why the two might misalign, and why one might need to question the optimality of the Scientific Method in Economics. The most apparent evidence of such a divide between the objectives for economics from inside and out, is the corresponding divide between economic methodologists and economic theorists:

renowned theorists regard methodologists as aloof (Solow, 2002), while methodologists believe their recommendations are simply ignored (Frey, 2001).

Given that Macroeconomics has had questionable success in predicting either of its principal topics, economic fluctuations and growth, we see a remarkable obeisance to the more than century old core economic theory. This conservatism could be easily attributed to the rational self-interest that is often assumed in economic theory: scientists whose human capital is concentrated in the orthodoxy (the mathematics of optimisation on convex sets, say, for Economics), and would find it hard to adapt, have every incentive to promote conservative interpretations of evidence against that orthodoxy. But this is an objectionably cynical perspective, because anyone working in Science will find it at odds with the way they view themselves and their colleagues. Any science is an inherently social enterprise, and so naturally we might look instead to Social Psychology for help in modelling this conservatism. A tool here is the idea of *cognitive dissonance*, that individuals feel a drive to reconcile all their ideas and behaviours in the same way that they are driven to eat or sleep. The author of Cognitive Dissonance, Festinger (1962), gives the basics in terms of a smoker who is presented with evidence that smoking is bad for their health, and I will do the same. The smoker feels discomfort, akin to hunger, because they believe that they rationally choose to smoke but also that they love life and their own health, which are mutually exclusive ideas. A natural response would be to stop smoking and remove that disquiet by changing the behaviour. But, if that change is itself too painful, because of their nicotine addiction, then an alternative response is to change the belief that smoking is bad for their health, or the belief that they care about prolonging their life. The smoker may then decide that the evidence is ill founded and choose evidence of their own, or regard their later life as more miserable and less worth prolonging. Applied to Macroeconomics, or Science more generally, it is easy to imagine a similar response from a researcher confronted with evidence or an argument against their paradigm, that a neutral outsider would consider strong evidence against it. If that researcher has a world view deeply integrated with that paradigm (the inefficiency of central planning, say, or the primacy of indi-

vidual choice) then we might expect them to discount that evidence/argument in order to avoid the dissonant belief that the way in which they understand much of their world will not square with it. In this way, the institutions of Science can find a pressure from within towards conservatism, without the scientists even being aware of it. When a system of thought has been so persistent that it is used outside academia, like meaningfully aggregated supply and demand or the idea of a God active on the physical world, then there will be pressures from the Cognitive Dissonance of individuals outside the institutions too. In this way the evolutionary environment of a Scientific institution might well favour conservatism over description or prediction of phenomena because, for example, those apportioning the funding only understand research described in familiar terms. There is evidence that economists are more subject to such bias in their research, than other scientists, with a significantly higher bias towards the publication of positive results than the biological and physical sciences. This suggests either that Economists have better intuition than scientists elsewhere or, more likely, that they are somehow promoting positive results through their interpretation of evidence (Fanelli, 2010).

Conservatism need not be driven solely by individual preferences, it is argued to be an inevitable feature of the prevailing method in Science, that of falsification. The Duhem-Quine Thesis, a famous argument in the Philosophy of Science, points out that rejection of a particular model is not a rejection of all the assumptions of that model but only of that combination of assumptions. Because individual assumptions must be tested together in a complete model, rather than in isolation, this means that some theoretical assumptions can be part of many rejected models without ever being abandoned, being protected by an ever changing cloud of auxiliary hypotheses of almost limitless supply (see Curd and Cover (1998)). An example of the Duhem-Quine Thesis accessible to any economist would be the Efficient Markets Hypothesis. Broadly, it states that market prices will reflect all available information determining the value of a good. Unfortunately, we do not know what the value of that good should be otherwise. For this reason, we cannot test the Efficient Market Hypothesis alone, but can only test the

joint hypothesis of whether market prices reflect the good's price according to some pricing model —a problem known as Market Indeterminacy (Fama, 1991; McGovern, 2006).

As well as rational cause for concern over the institutional structure of Science generally, there are also criticisms focussed on the institutions of Economics in particular. It is argued explicitly by Colander (2010) that the institutional structure of the Economics profession does not promote real-world fit, and therefore will not drive methodology to promote this:

> The Economics profession is primarily an academic profession, which sees itself as predominantly concerned with the science of economics, not with hands on applied policy advice. The institutional structure of the academic Economics profession is not structured to reward economists for the correctness of their real-world predictions, nor for their understanding of the real economy. Within academia, hands-on research, in which the researcher gains first-hand knowledge through working in the institutions he or she is studying, generally receives little reward.

Indeed, in his reflection on institutional causes of the 2008 financial crisis, Colander argues that universities and funding do not support a "representative researcher" thinking about all aspects of economic research, including the limitations of models for policy. Researchers are rather specialists, focussed only on specific questions within a research program, because this is how they will most likely progress up the career ladder. Crucially, there are no incentives to think about the merits of that research program as a whole, nor to leave it even if they thought it unlikely to succeed. For these reasons, Colander argues that Dynamic Stochastic General Equilibrium became predominant in Macroeconomics not because it offered the best chances of predictive success, but because it was easy to make incremental progress on it and build a profession around that progress. He talks of a profession that seeks to dot i's and cross t's rather than "...explore multiple models in a creative exploratory way". His diagnosis is of a peer

review process which is too restricted to individuals from a small set of universities and traditions, with nothing to stop them limiting the publication of other universities and traditions: "...too narrow a genetic pool, and too much inbreeding". Obviously, a review system promoting minor modifications to existing models is particularly prone to the Duhem-Quine Thesis, with the inherent problems described above.

There is a dichotomy in software engineering of the Cathedral and the Bazaar, differentiating projects developed by a small closed group from open source projects to which anyone can *try* to contribute. The successes of the latter are attributed to the fact that more pairs of eyes are more likely to quickly find errors. Now it may be that with the now trivial costs of distributing research, a Bazaar-model institution could be established that still promoted research and researchers that were more central to the scientific endeavour. The integer search ranking of research, rather than the binary published/unpublished, would then need to reflect its quality. Basing this and academic rankings on eigenvector centrality (as per Google's page rank), in the citations network, might be enough to marginalise crackpot researchers while still allowing promising heterodox ideas to come in from the periphery. Nevertheless, in the absence of such an innovation, the Cathedral and closed peer review seem the best system we have for improving and filtering research; we must accept its limitations and make a deliberate intervention on methodology.

2.2.1 Reductionism and Holism

Apart from the institutional features discussed in the previous subsection, there is a problem peculiar to current Macroeconomic theory, in that it can be, and is, argued to represent macroeconomic phenomena in two very different ways: as a representation of aggregate phenomena only, immune to criticism of theoretical assumptions; OR, as a representation based on the disaggregated components, immune to criticism of its empirical performance. This provides an additional line of defence for macroeconomists because, whenever their work is criticised from one perspective, they can simply

claim its relevance comes from the other. My criticism of the DSGE paradigm will be more complete than those elsewhere because it assesses the models from both perspectives, and finds them wanting from both. For simplicity I will henceforth refer to these perspectives with the terms *Holism* and *Reductionism*.

Holism and Reductionism are terms used in subtly different ways by different authors. Here, I choose the more natural use for the term Holism as the description of an object that does not take into account the behaviour of its components, considering only the *whole* object. Note that this is not a normative statement on what description should be adopted, only an attempt to classify some descriptions as independent of how components are thought to behave. For example, a holistic description of a bicycle tells us that turning the pedals makes the bike move, but it is agnostic about how: our model is a *black box*. As a complement to my definition of Holism, Reductionism is then the practice of describing an object's behaviour in terms of the behaviour of its components. For the bicycle, this means having a description of how the links of the chain articulate and engage with the gear such that the turning of the pedal causes the rotation of the rear wheel.

To understand the merit of Holism, note that no child needs to understand the mechanism of the chain and gears to start riding a bike; moreover, no adult can understand how to ride a bike based only an understanding of quantum mechanics. Reductionism is the pursuit of a description in terms of components' properties. To understand its merits, note that one could not anticipate the need to oil a chain without understanding the properties of its constituent metal, nor the problems of friction for the articulation of its links and the propulsion of the bike. In more general terms: Reductionism may needlessly complicate a description, or make description infeasible; Holism may deny relevant information from someone who wishes to predict. The following criticism of current microfounded macroeconomics stresses that its models fail from both these perspectives, and therefore fail in general.

Reductionism has a long history in the Social Sciences, where it is seen in the special

case known as *Methodological Individualsim*. Methodological Individualism holds that social phenomena can only be understood with a model of how they arise from the behaviour of the individuals involved. That is, it is Reductionism where the object of interest is a social phenomenon, and the components of interest are individual people. Although the term has much currency in the Economics literature, my general aim in this chapter is to discuss Economics in more universal methodological terms. I will therefore use the term Reductionism extensively, because it makes clear that there is a general case for comparison and that the Social Sciences do not merit exceptional (and overly leniant) methodology. It should be noted as well that the Economy can be, and is, described in terms of other components besides individuals —most obviously the goods, labour, and credit, markets of traditional Keynesian Macroeconomics.

Why should there be much difference between the holistic and reductionist description of an economy? The economy simply behaves like a market, which in turn behaves like an agent, right? Although much of modern Macro rests on this assumption, it is generally invalid. It has been long established that many relevant aggregations in Science are *non-linear*, in the sense that properties of the aggregate are not simply the sum of the components' properties. This was realised in Economics at least as far back as Hayek:

> The overall order of actions in a group is... more than the totality of regularities observable in the actions of the individuals and cannot be wholly reduced to them... a whole is more than the mere sum of its parts but presupposes also that these elements are related to each other in a particular manner.

Indeed, Hayek didn't argue for the invisible hand from a reductionist perspective at all, but from the holistic perspective of evolution driving social institutions towards optimality[1]. What prevents linear aggregation? Put simply, the non-trivial interaction

[1]Of course here he fell foul of non-linearity himself, neglecting that the environment to which a society's culture adapts is partly made up of other societies; the more recent concept of an *evolutionary landscape* would have been more useful (Bak, 1996).

of the components: the effects of any influence components have directly upon one another will not be captured if one ceases to consider the components and only the aggregate. The approach taken in the DSGE literature, assuming that the components only interact trivially, is sometimes known as Naive Reductionism or the *Fallacy of Composition*. In section 2.3 I will present several prominent results on exactly how the relationships in DSGE models, along with the agents themselves, are unrealistic.

2.2.2 Friedman's Instrumentalism as Holism

How could assumptions have been retained despite being shown to be unrealistic? In economics, Friedman's methodological writing (Friedman, 1953), which is seminal among the orthodox community, is often taken as justification for putting canonical economic theory before observations of actual microscopic components. This is partly a paraphrasing of Friedman's famous *F-twist*: "...the more significant the theory, the more unrealistic the assumptions...". Mäki (2003) shows that Friedman can be taken to argue any one of many different philosophical positions during his fairly incoherent text, which should be reason enough not to blindly follow its recommendations. However, Friedman himself approved the formal interpretation laid out by Boland (1979), of the F-twist as *Instrumentalism* —which was conceived in physics as a response to the contradictions between Einsteinian and Quantum Physics. This amounts to saying that the model is an account only of the phenomenon of study —the photoelectric effect, say. Hence, those other objects appearing in the model do not correspond to *real* objects observable by any other means; they are merely *instruments* in the description —i.e. photons are not real objects but a descriptive device for the discreet energy levels of the photoelectric effect. Running counter to Instrumentalism is *Realism*, which commits to the idea that models represent an objective reality —so Photons must be real objects if they appear in the description. Abandoning Realism is all well and fine when all we seek to do is to describe a particular phenomenon. I will discuss Realism and its counter-intuitive pitfalls more deeply in section 2.6. When we seek to

relate one phenomenon to others through Reductionism — say macroeconomic volatility to microscopic decision making — Instrumentalism clearly runs counter to our intent, because the observability of the other phenomena is central to our objective. So, if we embrace Friedman's as-if methodology, in the General Equilibrium context, we are embracing Holism and saying its "agents" are mere instruments and agents in name only.

If even a good description of past macroeconomic behaviour doesn't seem to work outside the sample for which it was chosen (into the future or other policy conditions), we might hope for more luck by finding a description of its components that does, and then using that description to find the behaviour of their aggregate. In this way we could see Reductionism compensating for an inadequate sample of observations of one phenomenon, by including information about related phenomena. As just explained, the Holistic interpretation of Friedman does not allow this, as we need microfoundations that are actually descriptions of the economy's components. But how good must these descriptions be? Which components matter? Obviously we cannot implement a reductionist model of the economy if we must describe every person in Britain, and the minutae of their relationships. The obvious answer is that we can omit only those details that don't stop our model reproducing macroeconomic behaviour, but now we start to sound like a different Friedman. As Maki (2000) argues, we don't have to abandon Instrumentalism and take a stance on Realism, rather we must find some sense of the "realisticness" of assumptions. To clarify this microfoundation-friendly take on the "as-if" maxim, we can adopt the typology of Musgrave (1981): *negligibility assumptions* involve the omission of component details that have no effect on the aggregate behaviour; *domain assumptions* involve abstractions from detail that have no effect for certain states of the aggregate; while, *heuristic assumptions* involve abstraction of details that change the behaviour of the aggregate. If a model is to be used generally, then it must have only negligibility assumptions, and so any reductionist model should meet this criterion. Musgrave makes clear that a model involving any heuristic assumptions at all is not appropriate for prediction. My criticism in the next section, of current

microfounded Macroeconomics from the reductionist perspective, will revolve around showing that not just one, but many, of the assumptions central to General Equilibrium are heuristic in nature.

2.3 DSGE as Reductionism

To reprise the previous section, if we can't find a reliable description of some phenomenon, an alternative might be to identify it with a collection of components that we can describe predictably. It is in this spirit that one might interpret the argument of Lucas (1976), in the paper credited with starting the microfoundations revolution in Macroeconomics. Lucas wrote this in the context of a then orthodoxy that was reductionist in a very different way. The Keynesian Macroeconomics of the time tried to reduce the economy to aggregate markets for goods, labour, and investment —described with the traditional intersecting supply and demand curves. Lucas's argument was in fact no more sophisticated than to assume the truth of microeconomics' General Equilibrium models and condemn Keynesian Econometrics for not using them. His claim was that the aggregate markets in these models might be expected to change qualitatively as agent behaviour responded to new policy conditions, and that robust models had to be parametrised by production technology and the preferences of consumers, which he took to be uninfluenced by policy. I don't think it controversial to take this as an endorsement of reductionism, particularly when Lucas (1991) talks in terms of describing micro and macro phenomena with the same theory:

> The most interesting recent developments in macroeconomic theory seem to me to be describable as the reincorporation of aggregative problems such as inflation and the business cycle within the general framework of "microeconomic" theory. If these developments succeed, the term 'macroeconomic; will simply disappear from use and the modifier micro will become superfluous. We will simply speak, as did Smith, Ricardo,

> Marshall and Walras, of economic theory. If we are honest, we will have to face the fact that at any given time there will be phenomena that are well-understood from the point of view of the economic theory we have and other phenomena that are not. We will be tempted I am sure, to relieve the discomfort induced by discrepancies between theory and facts by saying that the ill-understood facts are the province of some other, different kind of economic theory. Keynesian 'macroeconomics' was, I think, a surrender...to this temptation. It led to the abandonment, for a class of problems of great importance, of the use of the only 'engine for the discovery of truth' that we have in economics.

In this passage Lucas shows his reductionist bent, but seems oblivious to the Fallacy of Composition, treating it as unproblematic to think of many interacting markets in exactly the same terms we think of those micro components. The microeconomic theory to which he refers is that built around the concept of General Equilibrium, and it is the naive reductionism of General Equilibrium rather than microfoundations generally that is challenged in this section. Comprehensive surveys of the negative results for General Equilibrium exist (for example see Ackerman (2001); Rizvi (1997)), and I will mostly recapitulate here with only a few refinements and extra details.

2.3.1 Equilibrium

It is an old idea in Economics that market forces produce an *equilibrium* price vector that leaves no participants with incentives to change their behaviour. A common historical narrative is that the early neoclassical economists, like Walras and Pareto, emulated the mathematics of static equilibrium, and the maximisation of utility (minimisation of potential energy), from the physics of the time(Ackerman, 2001). Meanwhile, in the early 20th century, mathematics became infatuated with the axiomatic formulation of the Bourbaki group. Gerard Debreu was educated in this context, and

so his formalisation of the market equilibrium concept put no small emphasis on a minimal set of fundamental axioms (Kirman, 2006). The Arrow-Debreu formalisation of General Equilibrium that emerged was lauded for its simple clarity, reducing the economy to stark descriptions of many producer and consumer agents who produced and consumed in response to prices that automatically took on a value that left no excess demand. This automatic equilibration was described as the action of some *Auctioneer* in the Walrasian tradition, perhaps supposed to implicitly represent some other market process.

The Arrow-Debreu framework gained momentum so quickly that by the time negative results began to emerge, nothing could dampen the profession's enthusiasm. Sonnenschein (1972), Mantel (1974), and Debreu (1974) himself, found that within the Arrow-Debreu framework applied to goods exchange the uniqueness of an equilibrium cannot be assumed when the number of agents exceeds the number of goods, unless patently unrealistic assumptions are made about preferences[2] — henceforth known as the SMD results. More specifically, the results showed that the function representing aggregated excess demand need not have the properties that ensured a unique equilibrium, even if those properties had been assumed for the individual agents. As explained in the previous section, this non-linearity is typical of systems with many interacting components, so it should hardly have been surprising. Indeed, the equilibrium sets of prices (attractors) that ensured no excess demand could allow oscillating or even apparently random prices. Although this extreme behaviour is seen only for a trivial set of initial endowments, the more general case of locally unique equilibria would permit all kinds of dynamics when perturbed by the shocks of a DSGE model. Nevertheless, these results did not stop the Arrow-Debreu General Equilibrium from being integrated into Macroeconomics as a response to the Lucas (1976) Critique.

The SMD results mean that simply assuming a unique equilibrium exists, without specifying some demonstrably realistic preferences that guarantee it, is a Heuristic As-

[2]Specifically, the property of Gross Substitutability across goods, which specifies that when the price of one good rises the demand for every other good should fall (Mas-Colell et al., 1995).

sumption in the typology of Musgrave (1981) and renders the model invalid for prediction. Specifically, the assumption blinds us to all kinds of dynamic behaviour just as possible at equilibrium as a stable fixed price. Moreover, comparative statics are compromised even at a locally unique equilibrium, because any kind of discontinuous disturbance, like those assumed in macroeconomic models, could knock the system into the basin of attraction of another equilibrium. Indeed, Kemp-Benedict (2012) shows that general equilibrium's assumptions make it comparable to a topological field theory, which in turn implies the possibility of abrupt transitions between equilibria.

Much of the DSGE literature dodges the SMD result by simply assuming that the consumers in an economy can be replaced by a single *Representative Agent* responding to aggregate variables. This practice is clearly another heuristic assumption, as relaxing the abstraction to the more realistic multiple agents leads to different general model behaviour. Given that the SMD results allow oscillatory behaviour at equilibrium, a model of a rational decision maker is no more representative of General Equilibrium than is a model of a pendulum. But, the limitations of condensing all the economy's agents to a single representative do not end there. Kirman (1992) extends the SMD result to economies arbitrarily close to having homogeneous agents. He also reproduces a simple example in which the preference of a representative agent over two possible scenarios is the opposite of the preference of the two agents they represent.

Of course, the main concern of the DSGE literature in Macroeconomics is not goods markets, but rather the intertemporal allocation of investment resources, which is supposedly influenced by monetary policy. Here, Krusell et al. (1998) show that heterogeneity does not radically alter behaviour from the representative agent case, because long-run planning makes the Marginal Propensity to Consume effectively constant — allowing linear aggregation. Nevertheless, Carroll (2000) shows that Krusell et al. (1998) employs heuristic assumptions, because those generating a realistic income distribution also make the representative agent a poor approximation. This is of little consequence, however, because the SMD results for the goods market are not orthogonal

to these intertemporal decisions anyway: the choice between consuming now or in the future will be interrelated with the leisure-consumption trade-off in the present, which is subject to the SMD results, and faces conditions that could jump around unpredictably.

2.3.2 Absent Dynamics

While the historical narrative attributes the equilibrium concept in Economics to emulation of Physics, Economics did not stay in step for long. Physics went on to include Hamiltonian dynamics, with a measurable quantity conserved even out of equilibrium, in the form of energy. Economics' utility minimisation could not be meaningfully extended to conservation in a dynamic setting, and so this new maths could not be borrowed (Mirowski, 1990). If there was any will to develop a dynamic theory of Economics after Keynes's emphasis on disequilibrium, it had to die with the Arrow-Debreu framework which continued to marginalise dynamics away from equilibria. The Walrasian Auctioneer was a black box that hid whatever process was responsible for arriving at those equilibria, and one can understand its appeal; ignoring dynamics saved economists the burden of modelling all the complexities of actual market exchange, and let them skip straight to the simpler ideal outcome of that exchange. Unfortunately, without explicit dynamics equilibria are not what economists treat them as, because it is dynamics that would determine whether the economy moved towards that equilibrium or away from it. Hence, there is again a heuristic assumption in use when economists employ even comparative statics, because they do not know whether the economy will then diverge from the equilibrium they are studying.

There have been attempts to introduce dynamics into the general equilibrium story, first by making explicit the common intuition that prices change in response to excess demand. It can be shown that the simplest of such rules does not guarantee convergence, but of more concern is the informational requirements of such a process. This is because the Auctioneer must represent some actual physical process of interacting agents

with finite information processing capacity. Saari (1985) shows that the Walrasian Auctioneer cannot be replaced by any algorithm, based on just excess demand, that does not require infinite information to always converge. This is because either the process must be continuous, and hence process a continuum of information, or must have access to an infinite number of derivatives of the excess demand function if discrete. That is, no realistic mechanism could find a vector of prices by responding only to excess demand, in order to eliminate that excess.

Other processes can be considered that allow global convergence, by also including price information. But, Jean-Jacques Herings (2002) has shown that all these established mechanisms depend on Browder's fixed point theorem. This requires the adjustment process to be continuous, again requiring infinite information processing capability. Hence, for economists to invoke the Walrasian Auctioneer is a heuristic assumption, because then relaxing the abstraction of infinite information processing changes the behaviour of the model.

2.3.3 New Keynesian Equilibrium

Those familiar with DSGE will find the last two subsections to be only a partial refutation, because only DSGE models in the Real Business Cycle tradition are based on the General Equilibrium concept as Arrow and Debreu described it. The New Keynesian DSGE models eschew the auctioneer and market clearing in favour of *Monopolistic Competition*, wherein firms choose prices for their differentiated goods so as to maximise profits. But, these changes to the supply side of the economy do not address the problems of the demand side captured by the SMD result: the individual firms still face aggregate demand curves that do not necessarily inherit the convexity of individual demand curves, making the uniqueness of a profit maximising equilibrium just as arbitrary an assumption. This result was established by Roberts and Sonnenschein (1977) before Monopolistic Competition had even been given formal microfoundations by Hart (1982), in the paper that laid the foundations for the New Keynesian models.

In fact, Hart explicitly stated that,

> Unfortunately, as is well-known by now, the Cournot-Nash assumption introduces a serious non existence problem (see Roberts and Sonnenschein [1977]). We avoid this by considering a particularly simple model, and by making a number of strong assumptions about demand functions. There is no doubt, however, that generalizing the model significantly could be hard. For this reason, the analysis presented here should be considered more as an extended example than as a general model.

These words of caution were not heeded, however, by the authors who elaborated on this structure. For example, Taylor (1979) simply assumes an *objective demand curve* in his seminal New Keynesian model with staggered price adjustments by firms.

In his more tractable revision of Taylor, Calvo (1983) replaces this objective demand curve with one based on the maximising behaviour of a set of *Sidauski agents*. But, such agents are assumed to be identical, thereby forcing past the SMD result with heuristic assumptions (Brock, 1974). So, in the New Keynesian case too, microfoundations do not actually represent Reductionism because they do not reflect the actual interaction of the descriptions of agents.

I am not aware of a published result for the New Keynesian framework that parallels the problems of the auctioneer's infinite information requirements. But, the New Keynesian firm's profit maximisation problem is much like the auctioneer's problem in that an equilibrium must be found based on an unknown excess demand. One might therefore expect exactly the same informational requirements, and count another criticism that carries to the setting of Monopolistic Competition.

2.3.4 Undefined Aggregate Stochasticity

The Stochastic part of a Dynamic Stochastic General Equilibrium model takes the form of aggregate disturbances to the whole economy. The reason that unknowns at the micro scale are not thought relevant to the unpredictability of aggregates is the *Law of Large Numbers*, the property of finite-variance distributions that sample averages approach the population mean with sample size. Here the problem for DSGE is not simply that parts of the model do not correspond to the actual components of the economy, but also that it is sometimes unclear just what components they actually correspond to. Chari et al. (2009) argue that the shocks in New Keynesian models, such as the exemplar of Smets and Wouters, do not all have obvious real world interpretations. For example, they note that the suggested distribution of random changes in the substitutability of workers cannot be realistic, and go on to show that this stochastic term could be interpreted in different ways with radically different policy implications.

Meanwhile Aoki and Yoshikawa (2012) question how well the mean of an economy's future time path describes its potential behaviour. That the mean should give us a good enough idea of how the economy can move depends on idiosyncratic shocks to its individual agents not having ramifications for the whole economy, that is on the Law of Large Numbers. However, the Law of Large Numbers depends on the variance of successively large samples growing more slowly than the mean. This would usually be described by a coefficient of variation that shrinks to zero with sample size. However, in at least one important class of stochastic processes, that Aoki considers more realistic, this property breaks down. The implication is that the time path of the mean of the macro system, such as that of the Representative Agent, will not adequately describe its actual path under certain stochastic structures. Empirically, this vanishing coefficient of variation property would clearly collapse if shocks' volatility grew fast enough with the variable of interest for an agent. This lends the criticism even more bite, given that firm size, and personal income seem to follow fat tailed distributions that may suggest conditions proportional to their size (Gatti, 2008).

A similar concern for the law of large numbers might be that if the idiosyncratic micro-shocks in a system propagate enough, even if of standard variance, then once again the variance of successively large aggregates need not grow more slowly than the mean. To make this criticism more concrete, one might consider the quite general model of Acemoglu et al. (2012). Acemoglu and Ozdaglar find that should the connectedness of one agent within an otherwise neoclassical economy remain proportional to that economy's size, in the limit, then the law of large numbers may only be applicable to much larger economies or may fail entirely. Further, they find that even where one agent's direct relationships are not too many, if they are indirectly related to sufficiently many other agents (say, if they are related to a just-above-average number of agents who are in turn all related to a just-above-average number of agents) then the law of large numbers again breaks down. They then find that the indirect links of sectors in the US supply network suggests that sectoral shocks might have significant aggregate effects.

2.3.5 Agents and Markets

Beyond the above problems with the way agents and markets are aggregated in the General Equilibrium tradition, there are also well known problems with the way both are represented. In line with the Neoclassical school, Agents' decision making is assumed to be an optimal response to their environment. After Samuelson this optimality was defined with respect to a *revealed preference* relation over the possible combinations of goods consumed, at specific times, in specific states of the world. Whereas the classical utility based foundation of demand was underdetermined, because it did not otherwise correspond to observable phenomena, revealed preference was theoretically operational because it could be measured by individuals' choices between offered combinations of goods. For the preference relation detected in this way to be equivalent to a demand curve, it had to satisfy the Strong Axiom of Revealed Preference by not allowing indifference between any two combinations of goods. Unfortunately,

strict preference between two combinations is observationally equivalent to indifference, in which case exactly the same combination might be chosen. This immediately introduces a behavioural assumption into the use of revealed preference, that is no less ad hoc than classical utility theory. Of course, in purely operational terms there is also the problem that observed choice behaviour doesn't cover nearly enough pairs of alternatives to give a usable preference relation (Rabin and Koszegi, 2007).

The unobservability of preferences obviously means that rationality as a description of individuals' behaviour can only be tested in combination with assumptions about the goals of that rational action. This brings us back to the Duhem-Quine Thesis, and attendant problems, which makes it difficult to assess whether or not rational agents are a realistic description of individuals at all. When consistency of preferences is assumed, rationality can be tested by giving experimental subjects the same choices presented in different ways. Here the evidence is generally against rational decision making: people struggle to perceive choices through the way they are presented, to predict their own preferences at later times, and to make short term decisions consistent with their long term choices (Rabin, 1998). One of the exciting things about decision theory, is the reflexivity between how the modelled decision makers build up knowledge and the method by which the researcher themself builds knowledge of their subject. In section 2.6 I will give some clue as to why describing decision makers as rational might not be appropriate, when they deal with decisions about anything but very regular phenomena — for instance, decisions involving an economy that is influenced by so many different phenomena we cannot yet predict.

Another way in which some have attempted to save General Equilibrium[3] is through using simpler descriptions of agents, but aggregated in a more sophisticated way. Both Hildenbrand (1983) and Grandmont (1987) try to derive a well behaved aggregate demand curve from the distributions of observable characteristics of consumers. To do so, both choose a more minimal description of individual agents' behaviours; Hildenbrand

[3] Albeit at the cost of losing the Fundamental Theorems of Welfare Economics, and hence any normative properties of market equilibrium.

only assumes Revealed Preference at the individual level, not the Strong Axiom, while Grandmont makes the even starker assumption that budget constraints are satisfied. In Hildenbrand's model, well behaved aggregate partial demands[4] are then implied by his cited distributions of consumers' responses to changing income[5]. But, this distribution is only observable from cross-sectional data if one assumes that individuals' choice behaviour is locally independent of their income level, which is not something that can be established from available data (Evstigneev et al., 1997). Of course, once again, infeasibility and the evidence against persistent preferences prevent Revealed Preference from being a good description of individual behaviour, and also undermine Hildenbrand's approach. Grandmont's approach is simpler in its assumptions about individuals, but its distributional assumptions are expressed in terms of a metric over preferences that cannot be applied to observable data, and so it also fails to employ descriptions of the economy's actual components.

While the description of agents employed in DSGE is a questionably negligible abstraction, the description of markets is of far less ambiguous quality. Douglas North famously wrote:

> It is a peculiar fact that the literature on economics...contains so little discussion of the central institution that underlies neoclassical economics — the market.

Indeed, it is argued that Walras had misunderstood the operation of the Paris bourse when he tried to relate it to his original discussion of market equilibrium. Kirman (2006) argues that the Arrow-Debreu framework scarcely resembles actual market activity, which involves information besides price signals and introspection. Of course, each of us will have plenty of experience of the bilateral nature of much economic activity, where prices are agreed for a single idiosyncratic exchange rather than being

[4]But note, not the full aggregate demand needed for an equilibrium (Grandmont, 1987).

[5]Specifically, he shows that a continuous decreasing density of the distribution implies that preference or indifference can be established for the aggregate over every pair of goods.

posted in some centralised marketplace. Indeed, even in supermarkets where prices are not open to bilateral negotiation, systems like loyalty cards allow prices to effectively be tailored to individual customers by introducing a second set of prices that interact with their spending in a complex way.

To describe the descriptions of agents and markets in DSGE models as based on heuristic assumptions would be to assume that they can be relaxed, in order to see the effect on model behaviour. In neither case is this possible while retaining any of the character of the model.

2.3.6 Failed Reductionism

From a policy design perspective, in light of the heuristic assumptions described above, reductionist DSGE models should be employed with great care. As deductive theory, these models' results will necessarily be tautologous to the set of (unrealistic) axioms on which they are based. This means that even when institutions like government are incorporated into a DSGE model, any roles of their policies that are already abstracted into assumed smoothness of markets will be neglected in the recommended allocation of resources to them: if we assume a Walrasian auctioneer, then of course we'll find that market regulation is unnecessary; meanwhile, even if we find a role for money in such models, if it doesn't conform to the actual behaviour of money in the economy there is no reason to think that policy recommendations will be relevant. Indeed, whenever modellers employ a heuristic *ceteris paribus* assumption, made for the sake of tractability, their results will only be relevant to policy questions that are wholly orthogonal to mechanisms they have assumed away (Boumans and Morgan, 2002).

To summarise, the arguments above do suggest that the assumptions of general equilibrium are not realistic in a meaningful sense, and this renders microfoundations based on general equilibrium obsolete: if the agents are not contextual approximates to real people, and nor are the relationships between them and the economy contextual ap-

proximates, then the model components share only their names with actual economic agents and there is no reason a-priori to think they will help describe economic activity. So even ignoring the unrealisticness of rational man as a description for individuals' behaviour, with any realistic link to the macroeconomy severed, the representative agent would still no more represent a consumer/producer than they would a child trading marbles in a playground somewhere. The latter would recommend a policy of finding the child and carefully managing their trades, and yet we don't pursue it!

The conclusion from this section should be that General Equilibrium Macroeconomics fails completely if interpreted from a reductionist perspective. A response to this might be to invoke Friedman's argument, that the macroeconomy might anyway behave "as-if" it were a DSGE model, even though its structure bears no relation. Invoked in this way, I established in section 2.2 that we are invoking Instrumentalism (if we are invoking anything meaningful). Because we have plenty of information about the microscopic objects in our models, Instrumentalism takes on a subtly different interpretation in Economics from that in Physics. In the latter it is effectively a statement about whether theories can contradict one another, but in economics it divorces macro theory from micro theory: it simply becomes Holism. Whether current Macroeconomics succeeds on these terms is a question I will pursue in the next section.

2.4 DSGE as Holism

Using Friedman's "as-if" argument to defend DSGE's lack of realisticness through Instrumentalism, is valid from a philosophical perspective, and so any criticism must be based on the models' performance from this perspective. Instrumentalism is usually discussed as an agnosticism over the realism vs anti-realism debate. In the context of Economics, no one would deny that micro theory tries to describe agents that actually exist. This then makes the significant implication of Instrumentalism that the "agents" of the macro model do not correspond to the agents of micro theory. This

Holism means that the only evaluation of DSGE models can be their power to describe macroeconomic phenomena.

Anyone subscribing to the Lucas critique might already object to this interpretation of DSGE, and they perhaps pre-empt some of my arguments. But, it should be noted that, interpreted any more widely than the context of Keynesian Econometrics, Lucas's critique misses the point that a holistic description might already include the response of policy makers in some way — just as one Taylor rule is now argued to capture all manner of complex responses from a central bank. If one *was* to look for a criticism of holistic DSGE in Lucas (1976), they could note that his attack also has bite against the VAR representations generally used to implement DSGE models: unstable vector-difference equation parameters are suggestive of a non-stationary process that cannot be described by a VAR, but could still be described with other holistic models. I will reprise this point after first describing some more current criticisms.

For the sake of clarity I will focus on a specific DSGE model, that of Smets and Wouters which is widely regarded as a successful example of New Keynesian Modelling, and is used for policy analysis by the European Central Bank (Smets and Wouters, 2007) — this model is described in detail in chapter 4 of this thesis. Because a Holistic model is only qualified by its performance on the macro data, we can dismiss DSGE as a holistic endeavour if the best of existing DSGE models fails: to argue that the promise of microfoundations justify continuing the program would be to invoke reductionism instead. The currency of this model makes it a fair candidate to stand trail for the whole research program.

The Smets-Wouters model is parametrised by Bayesian methods: the researchers represent their *prior knowledge* of the model's parameters with probability distributions, and then revise these distributions in light of the macroeconomic data using Bayes's Rule. This introduces subjectivity into the choice of parameters. As I explain in section 2.6.4, there is nothing wrong with this subjectivity per se. But, it does allow for potential (even unwitting) abuse, by introducing even more flexibility to the model for

researchers to a-theoretically bend behaviour to match data. What I am suggesting here is an extreme extension of the Duhem-Quine Thesis, where a large research program trying many auxiliary assumptions and prior distributions on only a few short data sets, can effectively act as if data-mining. In this case, forecasting or descriptive power driven by trial and error might seem like it was driven by the soundness of theoretical assumptions. This is similar to an argument used by Fama to challenge any evidence in the Efficient Markets Hypothesis debate (McGovern, 2006):

> With many clever researchers, on both sides of the efficiency fence, rummaging for forecasting variables, we are sure to find instances of "reliable" return predictability that are in fact spurious (1991, p. 1585)

Regardless of these concerns, the Smets-Wouters model is not successful at forecasting or describing macroeconomic data, at least by the sensible standards that I will explain. This argument only serves to explain why a DSGE model might match the performance of certain statistical models with more free parameters.

2.4.1 Forecasting

Foremost among the criticisms of Macroeconomics as a Holistic endeavour, quite naturally, are the persistent predictive failures of DSGE models, which tend to be outperformed by simpler econometric models on macroscopic data (Herbst and Schorfheide, 2012; Wickens, 2012; Edge and Gurkaynak, 2010).

Smets and Wouters themselves judge their model to be good at forecasting, because it produces forecasts that are comparable to Vector AutoRegressive processes with both maximum likelihood and Bayesian parameterisations — these are minimalistic descriptions of elements in a vector time series as a linear combinations of their preceding values. This good relative performance is echoed elsewhere (Edge and Gurkaynak, 2010), but there is a catch. Relative performance is a deceptive criterion, because

in absolute terms DSGE models, naive VARs, B-VARs, and traditional econometric models are all found to perform badly, both over the recent economic crisis (Wickens, 2012) and (even less forgivably) over the preceding calm of the Great Moderation (Edge and Gurkaynak, 2010). That is, no model succeeded in predicting many of the significant changes in GDP or inflation since 1992, predicted the great recession, nor predicted economic variables after the great recession. This poor performance is based on single variable forecasts, assessed using *root-mean-squared-error*, and Herbst and Schorfheide (2012) question whether the power of DSGE might lie in joint forecasting (calling on reductionist arguments). They construct a measure of this performance, but find the Smets-Wouters model does not offer a clear improvement over a three equation DSGE model — the size that might be taught in a classroom for its simplicity.

Despite the poor forecasting performance they find, neither Wickens (2012) nor Edge and Gurkaynak (2010) interpret this as grounds to dismiss the DSGE program. In fact, both point out that the models themselves predict their own inability to forecast well in the given conditions. That those conditions cover both calm and crisis between the two studies leads one to question whether this consistency is of any value. But, both papers argue that DSGE models should instead be assessed on other grounds — again, cognitive dissonance might help description here. Wickens (2012) argues that DSGE models are too much driven by unpredictable exogenous variables for them to forecast economic behaviour, and that they should therefore be assessed in-sample rather than out-of-sample. I will address DSGE models' descriptive performance in the next subsection.

Herbst and Schorfheide (2012) remark that DSGE, VAR, and B-VAR, are all outperformed by more sophisticated time series models. But this is not the only reason that DSGE's comparable performance to VAR and B-VAR is not a small victory for its proponents. I established in the previous section that DSGE does not work from a reductionist perspective, and that to escape this we have to appeal to Holism. The implication of a holistic perspective is that the model must be judged solely by its

performance on macroeconomic evidence. The "microfoundations" of DSGE are not, therefore, an advantage over the alternatives, they are a needless complication of the description of this data. This means that DSGE cannot be defended by its forecasting performance so long as Science values a neat, parsimonious, description of phenomena. I argue in section 2.6 that this is in fact the only criterion Science need employ.

2.4.2 Description

The response to DSGE's early predictive failures by its founding fathers, Lucas and Prescott, was to simply move the goalposts and ask instead only that models described recorded behaviour of the macroeconomy well (Evans and Honkapohja, 2005). This might seem like the response cognitive dissonance would predict to the empirical contradiction of a beloved theory, but it should be noted that the relaxation of standards from Maximum Likelihood prediction of paths to pattern description is in line with Hayek's beliefs about the modeling of complex systems — a view that is supported by the proponents of complexity science (Gatti et al., 2011). Indeed, Lucas had previously espoused such a viewpoint and was actually in agreement with authors as diametrically opposed as Herbert Simon (Boumans, 1997).

In developing a more sophisticated method for assessing DSGE models on these terms, Le et al. (2012) comment on the argument for the change away from forecasting:

> Supporters of DSGE models ... argued that these models were 'misspecified' or 'false' and so should not be tested by the usual econometric methods which would always reject them. Rather they should be evaluated by whether in simulation they could generate behaviour mimicing the 'stylised facts'. This mimicry could not be evaluated by statistical inference; rather it was an informal comparison.

This early method of comparing "stylised facts" generally amounted to comparison of statistics for the observed data with the same statistics' distributions in simulated

data. The statistics generally used were correlations between macro variables, and the Impulse Response Functions generated by introducing one of the orthogonal stochastic shocks, and tracking the subsequent behaviour of the model's variables. Le et al. (2010) argue that this approach is inadequate because it fails to consider the joint distribution of the statistics. This would allow a model to go unrejected because its correlations between output, inflation, and unemployment, were often close to the data, even if in any one simulation at least one of these correlations was very different. The proposed solution is a technique called Indirect Inference, which estimates the statistics on each of many simulations, and then condenses the joint distribution of those statistics into a single Wald statistic. This simulated Wald distribution can be compared to the Wald statistic for the observed macroeconomic data, and the model can be rejected if the observed Wald is very unlikely.

So how do DSGE models perform at describing the behaviour of macroeconomic variables? Again, I will discuss the exemplar Smets-Wouters model only, because of its currency and claim of success. Le et al. (2011) find that the model is thoroughly rejected by Indirect Inference, both in its original form and augmented with flexible prices. Only an arbitrary mixture of these two models is not rejected at the 99% significance level, and then only for a much shorter span of data (matching the "Great Moderation" period) and reduced sets of no more than three variables. Worse, in chapter 4 I find that for the single variable tests, where almost all of the model's 99% significance is seen, Indirect Inference has very little power to reject quite dramatically misspecified models.

Just as with forecasting, the performance is being assessed in Le et al. (2011) relative to a VAR, both because this is what the Smets-Wouters model is reduced to in practice and because a VAR is the statistic compared between observed and simulated data. So, once again we are confronted with the problem that, even had the description been good, it would necessarily be no better than the best fitting VAR description. Because we are speaking as Holists, we are accepting that the theoretical terms of the model

contribute nothing to its validity and we only care about its description of the macro data. Hence, our ultimate problem with the "stylised facts" method and its successors is that the best Holistic model is simply the "stylised fact" itself! In 2.6 I give the foundations for a method that would give a less tautological criterion for descriptive success, namely simple parsimony in describing the whole data set.

2.4.3 Policy Evaluation

Another response to the Lucas critique, was that of Sims (1980) to move away from theory and use *naive* models, such as unrestricted VARs, to describe the economy. This, too, amounts to Holism. The resounding criticism of this line of research has been that, without an explicit policy variable in the model, there is no way to judge the effects of interventions on the macroeconomy. That is, the model may provide a description of the macroeconomic variables given to it, but the interest rate here is determined by the other variables rather than an outside decision. The obvious policy instruments, then would be the innovations to monetary variables. But, when the innovation vector is correlated, the component affecting these variables directly is tied to contemporaneous innovations affecting other variables too, which one would not expect of policy interventions. Many transformations of the vector space would make the innovations orthogonal, and so it is not obvious which one isolates a *true* interest rate intervention. Supporters of DSGE therefore consider policy evaluation a fatal flaw of Simms's Holistic macroeconomic tradition.

Little do those invoking such a criticism realise that, if they also invoke "as-if" arguments with any meaning, DSGE models commit exactly the same error. Again, this is because the "as-if" argument supposes that the only important quality of a macroeconomic model is its description of the macroeconomic data in question, so that the features of the model named after agents and firms are no longer supposed to reflect the agents and firms of microeconomic theory or have the same relationships to policy. So in this case too, any identity is lost for the stochastic innovations to monetary policy

and there is no clear policy advice to be taken from the models.

2.4.4 Failed Holism

Conventional microfounded macroeconomic models do not predict or describe their subject phenomenon. This is all that can be concluded, given the above, and it means that the DSGE program fails from a holistic perspective. Part of the reason may be that DSGE models have been implemented in such a way that they could not address the time series properties Lucas (1976) originally decried; that DSGE models are generally reduced to VARs means that they cannot describe the shifting parameters of econometric difference equations, except through some exogenous process — like Markov switching. Of course, tacking on such an exogenous process to a DSGE model might improve its fit, but does not render the theoretical part any less superfluous. In chapters 3 and 4 of this thesis I will present a time series model that better describes the properties of macroeconomic time series, in the form of fractional integration.

I argue that the theoretical component of DSGE models is superfluous because: it fails at reductionism, and so does not allow information to be introduced from micro evidence; from a holistic perspective it only complicates the model without any gain in performance. Of course, for this argument to be a decisive reason to look elsewhere besides current Macroeconomics, I have to make clear that needless complication takes precedence over other concerns — like the intuition of some *great* theorist, say.

2.5 Describing Methodology

Before I can make prescriptions for methodology in Economics, one might ask that I first properly describe the current methodology. Section 2.2 suggested that economists themselves don't spend much time breaking methodology down past simple convention. But, that doesn't necessarily mean that those methodological conventions are

completely anarchic. In this section, I review some of the attempts to fit the Economic methodology *that is*, into some of the more traditional descriptive frameworks from the *Post-Positivist* Philosophy of Science: those of Thomas Kuhn and Imre Lakatos. These frameworks may be the most familiar to working Economists who have experience of methodology, and I choose them for this relatability. But, Methodologists in Economics moved away from these descriptions some time ago, to the even more relativistic positions of the *Sociology of Science* and to the *Pragmatism* that motivates my later prescriptivism (Hands, 2002).

The frameworks of Kuhn and Lakatos are mostly descriptive, rather than prescriptive like the methodology I will discuss in the next section. Both philosophers were students of Karl Popper, whose Falsification-centred methodology of *Critical Rationalism* is even better known among economists. But, Popper's ideas are generally regarded as a weak description even in Physics, for which they were conceived (Janssen, 1991). This weakness comes from their specificity, and they are so specific that they can also be interpreted as prescriptive. For this reason, I will deal with them in much more depth later, in section 2.6.

2.5.1 Kuhn's Paradigms and Revolutions

In short, Thomas Kuhn's model of Science is that a particular scientific community will be bound to a collection of theories, called its *paradigm*. Kuhn describes periods of *Normal Science* in which the peripheral theoretical details of a particular paradigm are fleshed out incrementally. He contrasts these with revolutionary periods of *Extraordinary Science* where the paradigm comes under doubt due to (vaguely defined) empirical failings, and entirely new paradigms can come to the fore (Caldwell, 2003).

Historians of Economics differ on what its central paradigm is, with the Classical, Neoclassical, and Kaynesian schools variously tried in the Kuhnian mould (Coats, 1969; Gordon, 1965). In terms of actual theoretical propositions, Gordon (1965) makes in-

dependent rational agents the paradigm of Economics. Several methodologists go on to specifically identify Arrow-Debreu General Equilibrium as the Kuhnian paradigm (Dow, 1981; Coats, 1969). It could be argued that the revolution between these two paradigms was the Marginalist revolution. But, this is dubious because the latter is simply a formalisation of the former. Authors rather identify the transition to Keynesian Economics as extraordinary science (Coats, 1969).

2.5.2 Lakatosian Scientific Research Programmes

Lakatos's ideas build on, and reconcile, those of Kuhn and Popper: Kuhn's paradigms become a *hard core* of immutable theory held by a particular community or *Scientific Research Program*, while the standard for empirical validity of peripheral theoretical details is explicitly referred to as Popper's *falsification*. Rather than a single Paradigm undergoing distinct phases of Normal and Extraordinary Science, Lakatos describes coexisting (and strangely independent) Scientific Research Programs. Over time the many possible peripheral theoretical details around an SRP's hard core are tested and variously rejected. In this way, Lakatos takes on the Duhem-Quine theorem by acknowledging that the hard core will not itself be confronted empirically. He even describes the peripheral theoretical assumptions as a *protective belt*. Lakatos also confronts the infinite number of possible peripheral assumptions, by identifying *positive heuristics* that scientists use to choose the next assumptions to test (Caldwell, 2003). As with Kuhn's paradigms, Lakatos's descriptions can be variously applied to schools, theories, and Economics as a whole (Janssen, 1991).

Lakatos does not prescribe how to do useful science, but he does tell us how to spot it. He describes *Progressive* SRPs, as those which provide new successful predictions of empirical evidence at a greater rate than paradoxical evidence is found (Caldwell, 2003). Otherwise, Lakatos considers an SRP to be *degenerative*. This distinction means that Lakatosian ideas could at least be used to judge whether the methodology of conventional Macroeconomics is indeed defective.

The literature's verdict is inconclusive, on whether current Macroeconomics is a progressive or regressive Scientific Research Program. Janssen (1991) recounts Weintraub's finding that General Equilibrium is progressive, but also gives detailed criticism of Weintraub's arguments, to leave it as a regressive SRP. Weintraub's Lakatosian characterisation gives General Equilibrium the following hard core assumptions:

HC1. there exist economic agents;

HC2. agents have preferences over outcomes;

HC3. agents independently optimize subject to constraints;

HC4. choices are made in interrelated markets;

HC5. agents have full relevant knowledge;

HC6. observable economic outcomes are coordinated, so they must be discussed with reference to equilibrium states.

This then leaves all the theory relating General Equilibrium to the macroeconomy as part of the protective belt, to be modified according to Weintraub's much less well defined positive heuristics. To get a more decisive verdict on whether General Equilibrium based Macroeconomics is progressive or regressive would need an extensive review of the entire DSGE literature. But, this undertaking might be of little value, given some widely recognised flaws with the Lakatosian model.

2.5.3 Limits of Post-Positivistic Descriptions

Ultimately, Janssen (1991) abandons the Lakatosian framework for describing General Equilibrium, and Economics more generally. He is not alone in considering the post-positivist models of science to be inadequate.

Like Popper's *Falsificationism*, the Kuhnian and Lakatosian frameworks are elegantly simple. But, like Popper's system, they are oversimplistic to the point of fatal vaguery. This is illustrated well by the flexible designation what parts of Economics count as *paradigms* and *Scientific Research Programs*.

More fatal for the Kuhnian account is the criticism brought by Weintraub (1979). He argues that Economics has developed as a continuous accumulation of knowledge, not punctuated by the abrupt changes of revolution. This is quite evident in the large overlap between classical and neoclassical "paradigms", and the integration of Keynesian thought into a single orthodoxy.

The further difficulty for Lakatosian descriptions of Economics comes from the applicability of his empirical criterion. Firstly, unidentifiable positive heuristics mean that any economic model requires ad hoc assumptions about how its variables relate to measurable empirical evidence, unlike the experiments of Physics for which Lakatos originally conceived his model. Where falsification is a matter of interpretation, the rate of successful new predictions is flexible how one chooses to interpret what models are saying empirically. Janssen (1991) also points out a deeper problem, in that accounting for the rate of successful prediction relative to falsification, across all the predictions of a research program, would involve a review of all research being undertaken in that program. When one considers that models making unsuccessful predictions are generally harder to publish, one realises that a proper accounting of whether a program is producing more good than bad predictions is impossible.

Even though post-positivist descriptions of the methodology in conventional macroeconomics are plainly not a recommendation of the DSGE program, the problems listed above make this far from the final decisive criticism needed. In the next section I address the vagueness of these frameworks, on how empirical evidence is handled, by looking only at operationalised approaches to handling empirical evidence.

2.6 Disciplining Methodology

The preceding sections should be persuasive to common sensibilities that modern macroeconomics has taken a misstep. But, the principal criticism, that macroeconomic models are not as parsimonious as other comparable descriptions, is only fatal if there is no more important criterion by which to select predictions. This is an expansive question that exposes the clay foundations of Science in general but, for those unconvinced by the above, I now attack these fundamental issues in pursuit of a decisive verdict. In simple terms the argument, of subsections 2.6.1 through 2.6.4, is that we have no technology by which to identify truths about a future objective reality, and we should instead embrace a self-consistent rule for subjective prediction. The rule I offer, in subsection 2.6.5 onwards, is that of assuming the persistent simplicity of the simplest description of everything we are instantaneously aware of — Occam's Razor applied only to subjective experience. I argue that this implies a Scientific Method along the same lines and, in particular, that we now have the language to follow this method formally in the form of the Minimum Description Length principle.

To discuss the goal and process of Economics the science we first need some basic terminology from the philosophy of science:

Deduction is a process on logical relations wherein we move from special instances to general properties: e.g. if we decide that all swans are birds (are a subset of the set of birds), and that a particular animal is a swan (they are a member of the set of swans), then we can deduce that the animal is a bird (they are also a member of the set of birds).

Abduction is the reverse process, of guessing a specific instance from a general class: e.g. if we think that only some birds are swans (the set of swans is a subset of the set of birds), and we have a specific bird (a member of the set of birds), then we can abduce that the bird is a swan (they may also be a member of the set of swans).

Induction is the assumption, if two things have many common features, that they will then both share other features: e.g. if all the things we have seen that have the shape and behaviour of a swan (are members of many of the same sets) are white (are also common members of another set), then we predict that any subsequent things with all these swan-like features will also be white (subsequent members of the observed sets will also be members of the unobserved set). Of course in the case of swans we would predict wrongly!

Ontology is a particular designation of what things can be described as existing: e.g. one might say that one person exists, but that the number one itself does not.

Epistemology is a particular designation of how one acquires knowledge and learns the ontology: e.g. one might be said (naively) to verify the truth of the statement "all swans will be white" by observing many instances where this statement has applied.

I only consider it worth discussing epistemologies that have been operationalised as some methodology, and this spares us a good deal of the more academic philosophy of science. That said, the epistemologies usually associated with both the mainstream statistical methodologies (*Frequentism* and *Bayesianism*) can both be used to save current macroeconomics. Broadly, this is the case because both epistemologies refuse to address the origin of the theoretical terms used in deduction, and leave the process producing those theoretical terms as a black box within individual researchers' minds. This stance, which I will refer to as *Rationalism*, ultimately allows abduction, through the intuition of researchers, to take priority over all other considerations like descriptive power and parsimony. In the following subsections I will dismantle each epistemology's Rationalism in turn, and then go on to describe a way of thinking that retains both statistical methodologies, successful scientific practice, and the common sense of my macroeconomics critique.

Induction lies at the heart of the increasing empiricism that has coincided with what many regard as social progress. But, for those who hold deduction above other modes

of inference, induction has the problem (known as *Hume's Problem*) that establishing its validity by the success of past predictions is a circular argument: Induction must assume consistency of phenomena over time, this is justified by consistency between previous time periods; but, to make future predictions based on past cases is itself an induction.

More practically, we must justify using induced models to recommend actions/policies, even though they can't tell us whether statements about policy outcomes are true. I can offer two rough arguments:

Consistency — our prescription applied to questions of inference should lead to prescribing itself — as, indeed, induction can because of the circularity described above.

Relevance — without inducing that our objectives (as referenced in past experience) will continue into the future, cost-benefit analyses of the future are impossible and prediction becomes irrelevant.

Ultimately, though, the simple induction described here is incomplete, because we must ask what exactly will be continuous through time. We might worry that what is expected to persist depends on how we have chosen to describe what exists now. To begin to answer this question, we need to be clearer about what we can know exists.

2.6.1 The Truth about Empirical Truth

It is well established that we can't justify induction in the pursuit of objective truth, but the discussion of Instrumentalism in section 2.2.2, which refuses to specify an ontology, provokes a revision of how we think about the goal of empirical Science generally. If we can describe Science without an objective reality then we can't describe its goal as working towards understanding objective reality. Some working scientists may well believe that this or some other is their end purpose, but for my purposes I will treat

Science as a quest for predictions of future observations, to aid decision making. The Realists' assumption of persistent real entities means that predictions about an object's behaviour are a necessary consequence of understanding it, because the formulation of a definition encompasses its regular behaviour. So, this definition is inclusive of Realists' activities. I also find it especially appropriate to Economics, where ultimately policy decisions will be made on the strength of models. Because people can also act without much consideration, and may do so on the basis of predictions of sorts, I will restrict my definition to conscious decision making. Hence,

Science — consists of social activities with the aim of predicting future empirical observations for the purpose of conscious decision making.

If we are discussing science in terms of knowing about future observations, then I should be more precise about what constitutes an observation. Henceforth, the term will mean one individual taking in empirical information. Because we are aware at most times of peripheral sensations to which we do not necessarily ascribe an object, and because on memorable occasions we misidentify objects from the sensations we have, I claim it is unquestionable that this empirical information only comes in the form of sensations:

Fundamental empirical truth — a continuum of sensory information is fundamental to what we identify in our stream of consciousness as discreet logical objects.

I am arguing that we describe these sensations as the objects we perceive, which are not themselves part of sensory information. Crucially, this means that the objects cannot be exactly identified with the sensory information, and we can deduce nothing from either about the other. The remaining components of one's stream of consciousness are the remembering/imagining of such logical objects, which I do not class as empirical truth:

Memory — is the information in our stream of consciousness, in the form of sets of discreet logical objects, that we identified as having happened previously.

Imagination — is the information in our stream of consciousness, in the form of sets of discreet logical objects, over which we feel we have control at any instant in time.

Conventionally, the word *truth* may be used as a blanket category for these informations, and this allows deductions between these classes. I don't mean to challenge that usage elsewhere, only to clarify here that the properties attached to "truth" can vary significantly. Finally, I also think it uncontroversial, after a little contemplation, to make the following definition:

Conscious Decision Making — is the processing of information in the stream of consciousness wherein action is chosen, through the sensations we can control, based on imagined information that we expect to become remembered information.

The purpose of these definitions is to give a clean classification of the informations involved in conscious decision making, and a precise definition of that process. Crucially, this lets me argue that it is a category error to treat the non-empirical information in the same way that we treat the objects we construct from empirical information:

Assertion 2.6.1 — because information in the stream of consciousness that refers to future events falls into the imagination category, we cannot deduce from it empirical information or remembered information, and the common designation of statements about the empirical future being *true* or not, in the same sense as statements about the past, are not logically valid.

That is, seeing that the swan is white is a distinct category of information from remembering a swan being white, and imagining that a future swan will be white. We cannot, therefore, deduce from seeing a white swan that imagined swans must be white, nor the

future sight of a white swan from an imagined swan being white: these things are fundamentally different experiences, not related through deductive logic. We must choose another rule by which to relate them.

What I am getting at is that a science which seeks to establish the *truth* of logical statements for conscious decision making, empirically, is inherently illogical, because the objects in a statement are a collection of different classes of which only one is directly related to empirical information. I am using my own language here, with the hope of making it accessible to another non-Philosopher with similar experience, but this rejection of truth status for statements about future empirical information is well established in the Philosophy of Science under the name of *Fallibilism*. It might seem that my emphasis on sense data as fundamental invokes the *Phenomenalism* of J.S. Mill or C.I. Lewis, but I will now argue that the above leads away from the Realism that has caused trouble for these ideas (see Bolender (1998) for a recent defence of this Phenomenalism).

If we recognise that statements about future observation cannot be described as true, then the statement that there is an objective reality is also meaningless, and we are led away from Realism to an agnostic position akin to Instrumentalism. This does not mean that behaving as if there is an objective reality will be useless, and I will reconcile these two in section 2.6.5. But, it does mean that we cannot treat objective reality as a given. An important consequence of this for methodology is that prescriptions for scientific activity cannot assume the existence of other scientists. Instead, a method must be consistent for both the scientific community, and an isolated individual treating the scientific community as a phenomenon to be predicted. Again, I will reconcile these needs in subsection 2.6.5.

Cognitive Dissonance (and common sense) would suggest that the reader's mind is now rebelling at the suggestion that there is not an objective reality. I hope that it will sooth a little to again clarify that:

> An objective reality may well be the best description of everything in

> each of our daily experience, and we should continue to behave as if there is an objective reality except in describing phenomena a long way from daily experience.

What I mean by "best" will become clear in subsection 2.6.5.

To summarise this subsection, we are left with three features a prescription for scientific activity should have:

Consistency — the prescription should prescribe itself.

Subjectivity — the prescription should not require an objective reality that is knowable through sense-data: it should be operational in terms of only our instantaneous stream of consciousness.

Objectivity — to be a prescription for Science, as a community, the subjective understanding must give us a concept of other people with which its behavioural prescriptions are compatible.

2.6.2 The Austrians' A-Priorism

An operational epistemology that unambiguously supports Realism, by simply rejecting empiricism, is the *A-Priorism* of the Austrian School of economics. It holds that objects are transcendental and inherently known to us, along with their future behaviour (Caldwell, 2003). That is, we already possess an ontology in some way. This denies uncertainty, but can be reconciled by the diluted position that there are some underlying mechanical relationships between such natural objects which are nevertheless deducible from fundamental truths that we inherently know. Indeed, even with the wane of the Austrian School (Caldwell, 2003) it is considered to still have currency in general (Hands, 2002). Nor is this position as unintuitive as it might at first seem; anyone reading about the repeated failures of economic policies could be forgiven for wondering whether formal empirical methods are too unwieldy to describe a

phenomenon as complex as an economy, or a person, and instead consider introspection as a basis for models. Indeed, the flight to abstract theory described by Colander (2010) might be explained the same way.

This epistemology allows descriptions of future behaviour (models) to be derived purely by introspection; something which has often left the Austrian school at odds with its empirical counterparts, but which cannot be rejected by the criteria of a different epistemology. Indeed, theoretical predictions are generally reconciled with observed counterexamples by either invoking some flaw in the deductive process, or some failure to identify the fundamental truths (Caldwell, 2003). Nevertheless, in admitting that fundamental truths can be misidentified we lose any decisive method by which to select them introspectively, and must return to empirical evidence as a means of confirming our inherent (but uncertain) knowledge of objects' mechanical relationships. This problem can be phrased in terms of the language of the previous section: possessing the ontology for logical objects does not allow deduction about empirical information. This empirical checking of theories derived by other means then looks a lot more like Karl Popper's critical rationalism, an epistemology widely subscribed to across the scientific community which is arguably operationalised in the statistical hypothesis testing that I will discuss next.

To put the Austrians' beliefs in context, consider the evidence against the model of rational utility maximisation described in section 2.3.5. The stereotypical Austrian methodology described above would pay no heed to such evidence, because the assumption of utility maximisation would be considered a self-evident fundamental truth. However, not all Austrians believe in utility maximisation: Caplan (1999) describes a schism in modern Austrianism between those comfortable with Neoclassical assumptions and those who reject them outright. This means that these two groups consider different sets of fundamental assumptions to be self-evident. My criticism of Austrianism stems from the question of how one establishes which of these fundamental truths should be believed.

To summarise, without this empirical extension, a-priorism fails the subjectivity and objectivity criteria, because it is not a complete prescription of activity.

2.6.3 Hypothesis Testing as an Incomplete Methodology

Similar interpretations are shared by both R.A. Fisher's hypothesis testing and the Falsificationism at the heart of Popper's epistemology. Both confront beliefs about an objective reality with some criticism based on empirical observation. For Popper the observation should show the belief to be absolutely false, which is a weak criterion against stochastic models that give finite probability to a continuum of statements. Hypothesis testing, on the other hand, only seeks to show that an observation is very unlikely given a certain belief. This unlikeliness is defined in terms of the Frequentist interpretation of probability: an objective reality is assumed to exist with the possibility of infinite repetition of any given circumstance; probability is then the relative frequency of a certain outcome over those infinite repetitions. A hypothesis is rejected, then, if observed phenomena would have a very low relative frequency were that hypothesis true.

Both kinds of falsification can also be interpreted as part of an evolutionary epistemology that slowly promotes scientific beliefs that are more consistent with evidence, by successively killing off those beliefs that are not. There is a difficulty here, however, in that neither could be expected to explore an infinite space of possible beliefs (relationships between variables) in finite time. So, there is an implicit assumption that these beliefs are abduced in such a way that they are more likely than random to be good predictors. It is the ambiguity in how these hypotheses are selected that introduces Rationalism into hypothesis testing, and Rationalism ultimately allows conventional microfoundations to persist while minor auxiliary assumptions are tried and discarded — as per the Duhem-Quine Thesis. Recall that the epistemology Rationalism assumes that new empirical knowledge can be deduced from imagination and memory, without sense data.

Confronting this incompleteness of the method then becomes a matter of replacing the black-box of Rationalism with an explicit prescription for suggesting hypotheses. Before this can be done, however, we need to be sure that the black-box which has so far helped scientific prediction can indeed be opened. Two arguments, that it cannot are common: first, the creation of hypotheses might be some mechanical process in the brain that we cannot replicate formally, due to the brain's complexity; second that, as the Austrians believe, we have access to some a-priori knowledge about future empirical information, not itself derived from empirical information.

If Rationalism is supposed to be some function of the brain that we cannot replicate formally, then the problem of subsection 2.6.1 still applies. That is, anything that your brain is able to discern from empirical evidence cannot be turned into a truth or falsehood about an objective world. This is of course at odds with a method that aims only to decisively reject false theories. There is also a problem in that Popperian philosophy would suggest that hypotheses be tested against new information, out of sample, in the way that General Relativity predicted gravity lensing. Clearly if a hypothesis is developed in the brain from sensory information, then that sensory information was also available to the conscious mind, and can't be tested out of sample. This then means that the black box is simply finding a good description of data, which is something that we might hope to replicate formally — more on this later. In the specific case of Macroeconomics, we see the converse problem in that data sets are so few, limited, and abstracted from what they represent, that the brain could be processing them only in the way that a computer might data mine — a black box that can definitely be opened.

Instead, one could argue that the black box of Rationalism cannot be opened because the brain has access to information that is not available to the senses. Opposing this position is the successful set of models in physics that limit the transfer of information at the macroscopic scale to local regions of space forward through time, which would make the method inconsistent with its own conclusions. In its favour is the attempt of the physicist Roger Penrose to revive the concept of platonic ideals, through quantum

mechanics in the brain. Penrose argues that there are structures within the brain capable of amplifying activity from the quantum scale, so that non-sensory information can effectively influence cognitive processes (Penrose and Gardner, 1999). However, Penrose' model has been widely rejected by both theorists and evidence (Tegmark, 2000), again making it inconsistent with its own conclusions.

Regardless of the provenance of the objects used for the deduction of new knowledge, there is a problem with Rationalism's limitation to deductive logic. Since the heyday of Critical Rationalism there has been a large amount of work on other logics, for instance: *Fuzzy Logic* acknowledges the ambiguity of set membership, because language is learned from experience and borderline cases exist (Zadeh, 1965); *Subjective Logic* acknowledges that set allocation may change over time, because Science is still developing our description of many phenomena (Jøsang, 2001); and *Constructivist Logic* denies the law of the excluded middle (the existence of set complements), because truth values cannot be established from empirical observation (Tieszen, 1998). These alternatives to classical deduction all raise the question of how it is justified as relevant to empirical experience, when the way in which we process logical objects isn't clear.

Of course, even were we to accept the primacy of deduction, and deduce knowledge about objects through decisive rejection, one still relies on induction to tell us that we are still observing the same object before we can apply that knowledge. This brings us right back to the problem of induction, and means that ultimately we are gaining nothing for all the extra complexity, in describing scientific activity, that Rationalism introduces. Again we are tempted toward a view of Science that doesn't seek to establish truths, but is at least self-consistent in its prescription for behaviour.

To put this discussion in the context of economic models. Much of the criticism based on the SMD results of section 2.3.1 has been from the perspective of whether General Equilibrium is *testable*. The hope would be that a testable theory of General Equilibrium could be supported empirically by showing that it is not incompatible with what we see the economy do. My argument here is that this focus is too narrow for several

reasons. First, this focus on whether or not a particular theory is testable stops shorts of asking how we arrive at testable theories, so it allows General Equilibrium to remain in focus perpetually in the hope that some subtle modification will correct this problem. Second, it is treated as a given that rejection of a model now will mean that the model is forever irrelevant, because a method based around natural laws is inflexible to changing phenomena. The economy at the micro level is forever changing as new technologies emerge; assumptions like all markets being centralised might be rejected now, but could become a reality in the future. This second argument might be seen as coming full circle, back to Austrian arguments against empirical tests (Caldwell, 2003):

> One reason that it is senseless to try to falsify economic theories is that there are no 'constants' in the social world equivalent to those encountered in the natural sciences.

As a final thought, Popper's promotion of Rationalism used Einstein's theory of General Relativity as an exemplar (Popper, 2002). But, though it was a radical departure from conventional physics, it should be remembered that General Relativity is only a particular implementation of Riemann Geometry, and that it was made possible by the evolution of a descriptive language for other subjects. Rather than relying on an inspired few to propose new theory from pure intuition, this suggests far more that science has proceeded by trying existing descriptions against new information. This is an idea that I will reprise after the next subsection. First we must consider instead the Bayesian epistemology, which isn't tied to the idea of objective truth, but instead describes subjective beliefs being revised in light of new information.

To summarise, falsification based on Rationalism fails the consistency criterion, because it has promoted a theory of brain function that doesn't allow non-sensory information about the world. Meanwhile, without Rationalism it fails the subjectivity and objectivity criteria, because it does not give a complete prescription.

2.6.4 Bayesianism as a Category Mistake

Bayesian statistical techniques have seen a great deal of interest in Macroeconomics over the past decade, and the library of associated techniques is as impossible to cover well here as that for Frequentist statistics. The fundamental idea, however, is quite simple with the researcher representing their states of knowledge, before and after some new observation, with probability distributions. Their degree of belief in a given model before the new observation, represented by a prior distribution ($\mathbf{P}(H)$), is informed by the likelihood of the observation under that model ($\mathbf{P}(X \mid H)$) to give their updated degree of belief in the model, or *posterior probability* ($\mathbf{P}(H \mid X)$). This is done according to *Bayes's Rule*,

$$\mathbf{P}(H \mid X) = \frac{\mathbf{P}(H)\mathbf{P}(X \mid H)}{\mathbf{P}(X)}$$

Crucially, this allows competing models of the world to be (partially) believed simultaneously, and when applied to parametric models can give rise to well behaved prior and posterior probability distributions over the parameters of the model.

A common criticism raised against Bayesian statistics is the subjectivity of this prior distribution. Such criticisms fail to understand the epistemology behind Bayesian probability: knowledge is inherently subjective, and the modification of the prior probability to give the posterior simply makes explicit the way in which it is assumed that knowledge grows. Those who have used Bayesian statistics in practice, however, will be familiar with a conceptual problem with the prior probability; in order to represent a state of ignorance probability theory would recommend a uniform prior, giving equal probability to all possible parameterisations (or, in the extreme, all models). To have some other prior would be to introduce non-empirical information, in the Rationalist way criticised above. But, a uniform prior over an infinite space would break the axiom of probability that the probability of *some* event happening be exactly one. Expressed another way, probability theory is capable of describing ignorance over an effectively

bounded set of possibilities, but not true ignorance where we do not want to commit to any assumptions about the future. This dichotomy of ignorance was recognised in Statistics as early as Savage, and in Economics even earlier by Knight (Binmore, 2007).

For an everyday example of the difference between these two states of ignorance, consider the difference between a roulette wheel and high energy particle accelerator. The roulette wheel exists to provide situations of unpredictability for gambling where we can price the various outcomes, so when we spin the wheel we must know that there are a particular set of numbers that can come up. We are completely ignorant about which number will come up, but we have a very effective description for any of the outcomes when it arises: we are dealing with *known unknowns*, or *Knightian Risk* as it is sometimes called. With the particle accelerator, however, we may have some theory to suggest outcomes but ultimately we do not know what the outcome will be because we have never experienced a particle collision at this energy before. Our models may describe some of what we observe, but ultimately there will be a lot that cannot be described easily and for which we will have to extend the models: we are dealing with *unknown unknowns*, or *Knightian Uncertainty*[6].

That Bayesian statistics conflates these two states of ignorance is a category error that ultimately undermines the whole epistemology. This is because Knightian uncertainty, as the lack of a language tying certain present observations to certain future observations, is isomorphic to the problem of induction. To discuss it in terms of events that might happen would therefore be the same as having knowledge of future observation. But, as I have noted already, we cannot logically derive knowledge of future observations from empirical sources.

If one believes that our own decision making faculties have evolved to be a near optimal approach to learning about our environment, then more evidence against the Bayesian

[6]This latter example may not seem particularly "everyday", but the very nature of Knightian Uncertainty makes describing examples difficult: such complete uncertainty is familiar to all of us, but because there is no regularity from which to predict we necessarily do not have language to describe the specific situations.

epistemology comes in the form of the *Ellsberg Paradox*. Here a decision maker is presented with the choice between two gambles, where one has known probabilities and the other does not. For example, one gamble might be winning ??1 million if a black ball is chosen from an urn with 100 black balls and 100 whites, while the second gamble was identical but with unknown numbers of black and white balls. Real decision makers confronted with this problem show a preference for the former gamble, that would suggest they didn't give equal prior probability to black and white draws in the latter — a violation of uniform prior probabilities. Further, offering them the same bets on white draws instead yields the same result, suggesting that their prior probability for *any* ball being drawn was less than one in the latter gamble — a violation of the definition of probability.

Extensions of probability theory exist to address the Ellsberg Paradox, and I find two of these notable. *Choquet Capacities* effectively place a lower bound on the probability of an event (Binmore, 2007), but this is an unsatisfactory solution to Knightian Uncertainty as it still describes some knowledge about the future. Subjective Logic, introduces a second dimension to probabilistic descriptions of future events, allowing for it to be more or less uncertain as well as more or less likely (Jøsang, 2001). Although this addresses some problems for describing decision making and could find fruitful application in Microeconomic models, it offers no prescription for learning about the world and does not redeem the Bayesian epistemology.

In the specific context of Macroeocnomics, Bayesian statistics is used for the empirical part of such prominent work as the Smets Wouters model described in section 2.4. The epistemological criticism here does not undermine the choice of priors used in Smets and Wouters (2007), but rather the meaning attributed to the whole exercise. Because Bayesian probabilities are not a good representation of subjective beliefs about phenomena, the prior and posterior distributions lose their meanings. The Minimum Description Length framework described in the next subsection allows for some Bayesian techniques to still be used despite this problem.

To summarise, Bayesianism fails the subjectivity and objectivity criteria, because it cannot effectively describe the state of ignorance at the start of the learning process.

2.6.5 Empiricism through Minimum Description Length

Having decided that deductive and probabilistic logic do not everywhere help us describe the Scientific Process, the indispensable component we are left with is induction. As I argued above, induction is at least self-consistent, because our experience generally suggests the continuity of our world as a best guess when in a state of ignorance. But, Nelson Goodman famously argued that implementing induction was entirely dependent on the language with which we chose to do so: he argued that we could invent a term *grue* that denoted something green up until the present but blue thereafter, and we would then struggle to choose inducing that grass would be green or grue in future. What is needed for induction alone to successfully describe Science is a rule for what language we induce from. This is where the Minimal Description Length Principle comes in.

That we should prefer a minimal description of phenomena is not a new idea. Aristotle et al. (1999) (book V) famously said "Nature operates in the shortest way possible", and the term Ockham's Razor has been used extensively in Science to describe various similar notions. It was with the advent of Information Theory that this idea began to take a more definite shape, in the work of Ray Solomonoff.

The length of a description of phenomenal data is dependent on the language, or languages, in which it is expressed. Solomonoff argued that the fundamental language should be the instructions needed to instruct a computer to reproduce the data from only basic operations. He showed that for large sets of data, the precise way in which this computer was programmed didn't matter and could be treated as if there was a *universal computer language*. This led to the concept of *Kolmogorov Complexity* as the length of this shortest universal computer language program that reproduced the data.

This idea does not lead to an automatic induction engine, because there is no way of finding the Kolmogorov Complexity of a given set of data, but it does give us an ideal on which a more practical inductive principle could be built. This was mainly done by Jorma Rissanen, who summarises the philosophy behind this Minimum Description Length principle in a way that satisfies the discussion earlier in this section on using the truth concept for predictions:

> We never want to make the false assumption that the observed data actually were generated by a distribution of some kind, say Gaussian, and then go on to analyse the consequences and make further deductions. Our deductions may be entertaining but quite irrelevant to the task at hand, namely, to learn useful properties from the data.

He believes that induction should be based on finding as much regularity as possible in data, and assuming that these regularities will persist. This leads to considering scientific models as languages, rather than representations of a knowable reality. So that a minimal description of data involves the shortest combination of model description and data description relative to that model. The ideas of theory and hypothesis as truth statements have no place in this inductive system, they are only parts of a model that is qualified by how much of the regularity it describes, rather than by any property of these theories (Grunwald, 2007).

Again, I might expect the scientific reader to find such a demotion of theory objectionable, given the predictive successes of Theoretical Science. As part of the way many models are constructed, successful theories are still very important in this system of thinking. I will elaborate on how shortly, but for now consider that a description of all the data one has will involve all the models used in that description, the more theory is common to these models then the shorter will be their description and the total description. In the extreme this generalisation becomes *unification*, where the same model is used for data from two different sources.

Another immediate complaint might be that in practice models involve error. The critical reader may already have questioned whether the minimal description of data is a trivial description that simply accepts massive error. This is not the case, because stochasticity is fully accounted for in MDL, but takes on a very different interpretation from the probability of Frequentist or Bayesian thinking. In short, the probability distribution determined by a minimum description length model is isomorphic to the distribution of description lengths for data in that model's language: it guides the choice of which data are described with short *words*, and which with long. That is, a high probability datum is given a very short description, while a low probability datum is given a long description. This allocation of *word* lengths means that when calculating the expected description length, the short *words* will get the most weight, and the long *words* the least, so that the expected description length is minimal. In this way a model cannot be chosen so as to be all error, because those errors have to be described in the model's language too, and high variance distributions will mean long descriptions.

The split with Frequentist and Bayesian thinking does not cost followers of MDL the entirety of mainstream statistical techniques. On the contrary, MDL actually encompasses many methods from both schools of thought and allows their comparison (Hansen and Yu, 1998), it only rejects the way they interpret probability. For instance, if we restrict our possible models to a particular parametric family, then implementing MDL simply becomes the Frequentist staple Maximum Likelihood estimation, because maximising the joint likelihood of the data means allocating the shortest possible description for that data (Hansen and Yu, 1998). Hypothesis testing would take on the interpretation of abandoning any model by which it would be too longwinded to describe the set of data more extreme than that observed, and I will reprise this point in chapter 4. Meanwhile, in more sophisticated MDL techniques we see the Bayesian marginal distribution, which is the average of models over the prior distribution: MDL allows us to decide even before observing data what model, relative to some set of models, will give us the best performance; one such is the Bayes Universal Model, which is the Bayesian marginal distribution with respect to some prior (Grunwald, 2007).

I have given a very cursory introduction here to the ideas underlying MDL, because going into technical depth is beyond the scope of this chapter. For the underlying mathematics, specifics of implementation, and a fuller discussion of common criticisms, the reader is directed to Grunwald (2007). For the remainder of this subsection I will argue that a more idealised *global* interpretation of MDL, applied to all the phenomenal information we have, answers the demands made earlier in this section of a sensible prescription for scientific activity. In turn, this will definitively reject current microfoundations in macroeconomics, by prioritising models' simplicity over the intuition of their creators.

Does MDL make sense, at least heuristically, as a prescription of subjective behaviour, based only on what we are certain of and without the assumption of an objective reality and other scientists? That is, how is it expressed in terms of the instantaneous conscious awareness of subsection 2.6.1? I argued there that the data we have at any one point is a combination of sense data, memory-object data, and imagined-object data. A global MDL, applied to all conscious thought, then becomes the process of instantaneously choosing the simplest way to fit together these different data: linking sense-data to thought objects in a minimal way. This means that where sense data doesn't fit well with remembered objects then new objects have to be conceived, and perhaps some old ones re-conceived: a description of conscious learning. Prediction with imagined-objects then naturally becomes induction, because minimising the description of conscious data prompts the imagined data to be steered into correspondence with the minimal description of memory and sense data, taking on the same set of objects and their behaviours. If we were to posit a similar process without conscious control, then Festinger's Cognitive Dissonance makes sense as the drive to minimise description length by reconciling objects and behaviours that are doing double duty describing the same data. In this subjective context we can give a special interpretation to the unification effect I mentioned above in the context of MDL for specific data sets. Say that I am getting the visual sense data that is best described as my being close to a swan with my hand extended, and that every time I experience this I also experience

the sense data of feeling feathers on a very large animal under my fingers. In this circumstance I can unify the visual and touch descriptions into a single model, “swan”, and have reduced the global description length.

As I argue in subsection 2.6.1, it is necessary for a scientific prescription to be valid from the subjective perspective of one’s stream of consciousness, but not sufficient. To be a prescription for the scientific community’s behaviour it must make sense for every individual in the community to interact in a way that makes the prescription a social activity too. Here MDL succeeds because of the unification effect just described. Interacting with other people (or, rather, the sense data we minimally describe as being other people) we can unify the sense data corresponding to verbal communication with the memory-objects in our stream of consciousness. As verbal communication takes on more and more subjective meaning in this way, we are forced to keep our global description length short by carrying over relationships between verbal objects into being relationships between our remembered objects, thereby communicating information about phenomena. To help understand this, consider an equivalent effect in the setting of a blind man and his cane. He comes to identify the sense data experienced directly through his hand manipulating objects with the sense data experienced indirectly as he manipulates objects with the cane. So even though the sensation in his hand is quite different when he touches a table with his hand and with his cane, he still comes to describe the table as a single object. I am suggesting that we can describe Science as a community of individuals using one another as canes.

This scientific-communication-as-a-cane idea addresses the debate between Realism and Instrumentalism raised in subsection 2.6.1. MDL doesn’t try to discover information about an objective reality, so is akin to Instrumentalism. But, although our description is always tentative, any memory-objects that can be unified with sense regularities from communicating with others can be treated as objective. So, global MDL is not simply a description of scientific social activity, but all our social learning. As long as we can unify much of our other sense and memory data with the communicated

sense data, we are justified in maintaining realist beliefs in daily life. But, if we don't everywhere assume objective reality, surely my earlier dichotomy of Holism and Reductionism from section 2.2 is also lost, along with the existence of real objects and their components? This is not the case if we extend the definition of *component* to data regularities. Consider that we may have identified a regularity, A, that we often recognise many instances of simultaneously, or within close temporal proximity. Consider that this collection of regularity As always coincides with the experience of a second regularity, B. We may then unify a collection of regularity As and regularity B, and label A as a *component* of B. That is, if we can derive B from A, then we no longer need the separate description B and have reduced the global description length.

With these ideas (and I, your cane) in hand, let us consider the greatest priority from subsection 2.6.1 for a prescription of scientific, or any other, behaviour: self-consistency, that the prescription leads to prescribing itself. As already mentioned, induction automatically has this in a loose way, because our experience tells us that it often works to assume that we will continue to experience the same sense-data. This will extend to MDL so long as MDL is a minimal description of past behaviour that led to predictive success. The criticism often raised against inductive inference as a description of successful Science, is that prediction of new phenomena seems to have been made by deduction from theories for related phenomena. Here we can apply my argument above about the role of theory in MDL: the *global* description length, includes all models, and so generalising a theory, or trying mathematics from elsewhere, to as yet unobserved phenomena is reducing the global description length. So, for example, when General Relativity predicted Gravity Lensing it was not a success of Einstein's abduction, but rather of reducing global description length by re-using Riemann geometry elsewhere. With the success of abductive Rationalism described, MDL becomes a minimal description of successful scientific activity relative to the alternatives, and hence satisfies its own criterion and is self-consistent. MDL therefore addresses all the criticisms of other operational scientific methods that I describe above and, crucially, makes parsimony of models a priority over the (thus far unsuccessful) intuition of theorists.

2.7 Conclusions for Moving Macroeconomics Forward

This chapter should have been a conclusive refutation of conventional microfounded Macroeconomics, for any reader. We are then left with the question of how to move Macroeconomics forward. I would argue that it should be considered a blank slate, onto which researchers should freely sketch diverse new ideas, and "...explore multiple models in a creative exploratory way", as Colander (2010) suggests.

But, in their explorations of new models, researchers should bear in mind several lessons produced by this critique:

Describing the macroeconomy with microfoundations is only introducing more empirical evidence from those micro components if it doesn't involve heuristic assumptions that change the behaviour of the aggregate model: sacrificing realisticness for tractability, may sacrifice everything

Describing the macroeconomy with microfoundations that employ heuristic assumptions makes the model a Holistic one, that must be judged solely on how well it describes the macroeconomy.

Parsimony is the crucial criterion for a model being a successful description, so beloved (but unsuccessful) economic theory should be left by the wayside, with useful descriptions from any other domain as the first candidates to replace them: interdisciplinism should replace conventionalism.

Parsimony can be operationalised through statistical methods based on the Minimum Description Length principle.

Followed fully, the final of these points would mean significant changes to the way econometrics is done. So, for the sake of keeping my modelling work accessible to the Economics community, I do not use MDL proper in the empirical parts of this thesis. In chapters 3 and 4 I develop empirical methods based on hypothesis testing,

to show that my models can succeed on the terms of conventional macroeconomics. I consider it a vital extension to also apply MDL proper. That said, hypothesis testing is still in keeping with the broad scientific principle I outline in section 2.6.5, because it establishes whether a particular model will provide a description of data that is of the *reasonable* length that corresponds to a particular significance level.

Given the small irregular datasets we have in Macroeconomics there is still a compelling argument for reductionism, as a way to use better descriptions of components to help describe the aggregate. For this reason, reductionism remains a promising approach, and in the next chapter I explore a new kind of microfoundations based on enforcement rather than exchange.

Bibliography

Acemoglu, D., Carvalho, V. M., Ozdaglar, A., and Tahbaz-Salehi, A. The network origins of aggregate fluctuations. *Econometrica*, 80(5):1977–2016, 09 2012. URL http://ideas.repec.org/a/ecm/emetrp/v80y2012i5p1977-2016.html.

Ackerman, F. Still dead after all these years: interpreting the failure of general equilibrium theory. *Journal of Economic Methodology*, 9(2):119–139, 2001. URL http://ideas.repec.org/a/taf/jecmet/v9y2001i2p119-139.html.

Aoki, M. and Yoshikawa, H. Non-self-averaging in macroeconomic models: a criticism of modern micro-founded macroeconomics. *Journal of Economic Interaction and Coordination*, 7(1):1–22, 2012. ISSN 1860-711X. doi: 10.1007/s11403-012-0088-3. URL http://dx.doi.org/10.1007/s11403-012-0088-3.

Aristotle, Waterfield, R., and Bostock, D. *Physics*. Oxford world's classics. Oxford University Press, 1999. ISBN 9780192835864. URL http://books.google.co.uk/books?id=QpGlDEJUDVAC.

Bak, P. *How nature works: the science of self-organized criticality*. Copernicus Series. Copernicus, 1996. ISBN 9780387947914. URL http://books.google.co.uk/books?id=e5XuAAAAMAAJ.

Binmore, K. Rational decisions in large worlds. *Annales d'Economie et de Statistique*, (86):25–41, 2007. URL http://ideas.repec.org/a/adr/anecst/y2007i86p04.html.

Boland, L. A critique of friedman's critics. *J. of Economic Literature*, 17(2):503–22, 1979.

Bolender, J. Factual phenomenalism: A supervenience theory. *Sorites*, 9(9):16–31, 1998.

Boumans, M. Lucas and artificial worlds. 29:63–88, 1997. URL http://ssrn.com/abstract=1434345.

Boumans, M. and Morgan, M. Ceteris paribus conditions: materiality and the application of economic theories. *Journal of Economic Methodology*, 8(1):11–26, 2002. URL http://ideas.repec.org/a/taf/jecmet/v8y2002i1p11-26.html.

Brock, W. A. Money and growth: The case of long run perfect foresight. *International Economic Review*, 15(3):750–77, October 1974. URL http://ideas.repec.org/a/ier/iecrev/v15y1974i3p750-77.html.

Caldwell, B. *Beyond Positivism*. Taylor & Francis, 2003. ISBN 9780203565520. URL http://books.google.co.uk/books?id=hrqLZsCOg4sC.

Calvo, G. A. Staggered prices in a utility-maximizing framework. *Journal of Monetary Economics*, 12(3):383–398, September 1983. URL http://ideas.repec.org/a/eee/moneco/v12y1983i3p383-398.html.

Caplan, B. The austrian search for realistic foundations. *Southern Economic Journal*, 65(4):823–838, April 1999. URL http://ideas.repec.org/a/sej/ancoec/v654y1999p823-838.html.

Carroll, C. Requiem for the representative consumer? aggregate implications of microeconomic consumption behavior. *American Economic Review*, 90(2):110–115, 2000. URL http://EconPapers.repec.org/RePEc:aea:aecrev:v:90:y:2000:i:2:p:110-115.

Chari, V. V., Kehoe, P. J., and McGrattan, E. R. New keynesian models: Not yet useful for policy analysis. *American Economic Journal: Macroeconomics*, 1(1):242–66, January 2009. URL http://ideas.repec.org/a/aea/aejmac/v1y2009i1p242-66.html.

Coats, A. W. Is there a structure of scientific revolutions in economics? *Kyklos*, 22 (2):289–296, 1969.

Colander, D. The economics profession, the financial crisis, and method. *Journal of Economic Methodology*, 17(4):419–427, 2010.

Curd, M. and Cover, J. *Philosophy of Science: The Central Issues*. Norton, 1998.

Debreu, G. Excess demand functions. *Journal of Mathematical Economics*, 1(1):15–21, March 1974. URL http://ideas.repec.org/a/eee/mateco/v1y1974i1p15-21.html.

Dow, S. C. Weintraub and wiles: the methodological basis of policy conflict. *Journal of Post Keynesian Economics*, pages 325–339, 1981.

Edge, R. M. and Gurkaynak, R. S. How useful are estimated dsge model forecasts for central bankers? *Brookings Papers on Economic Activity*, 41(2 (Fall)):209–259, 2010. URL http://ideas.repec.org/a/bin/bpeajo/v41y2010i2010-02p209-259.html.

Evans, G. W. and Honkapohja, S. An interview with thomas j. sargent. *Macroeconomic Dynamics*, 9(04):561–583, September 2005.

Evstigneev, I. V., Hildenbrand, W., and Jerison, M. Metonymy and cross-section demand. *Journal of Mathematical Economics*, 28(4):397–414, November 1997. URL http://ideas.repec.org/a/eee/mateco/v28y1997i4p397-414.html.

Fama, E. F. Efficient capital markets: Ii. *The Journal of Finance*, 46(5):1575–1617, 1991. ISSN 1540-6261. doi: 10.1111/j.1540-6261.1991.tb04636.x. URL http://dx.doi.org/10.1111/j.1540-6261.1991.tb04636.x.

Fanelli, D. 'positive' results increase down the hierarchy of the sciences. 5(4), 2010. doi: 10.1371/journal.pone.0010068.

Festinger, L. *A Theory of Cognitive Dissonance*. Mass communication series. Stanford University Press, 1962. ISBN 9780804709118. URL http://books.google.co.uk/books?id=voeQ-8CASacC.

Frey, B. S. Why economists disregard economic methodology. *Journal of Economic Methodology*, 8(1):41–47, March 2001.

Friedman, M. The methodology of positive economics. In *Essays in Positive Economics*, pages 3–43. University of Chicago Press, 1953.

Gatti, D. *Emergent macroeconomics: an agent-based approach to business fluctuations*. Springer, 2008. ISBN 9788847007246.

Gatti, D., Delli, G., Desiderio, S., Gaffeo, E., Cirillo, P., and Gallegati, M. *Macroeconomics from the Bottom-up*. Springer, 2011. ISBN 9788847019706.

Gordon, D. F. The role of the history of economic thought in the understanding of modern economic theory. *The American Economic Review*, pages 119–127, 1965.

Grandmont, J.-M. Distributions of preferences and the "law of demand". *Econometrica: Journal of the Econometric Society*, pages 155–161, 1987.

Grunwald, P. *The Minimum Description Length Principle*. Adaptive computation and machine learning. MIT Press, 2007. ISBN 9780262072816. URL http://books.google.co.uk/books?id=mbU6T7oUrBgC.

Hands, D. W. Economic methodology is dead - long live economic methodology: thirteen theses on the new economic methodology. *Journal of Economic Methodology*, 8(1):49–63, 2002. URL http://ideas.repec.org/a/taf/jecmet/v8y2002i1p49-63.html.

Hansen, M. H. and Yu, B. Model selection and the principle of minimum description length. *Journal of the American Statistical Association*, 96:746–774, 1998.

Hart, O. A model of imperfect competition with keynesian features. *The Quarterly Journal of Economics*, 97(1):109–38, February 1982. URL http://ideas.repec.org/a/tpr/qjecon/v97y1982i1p109-38.html.

Herbst, E. and Schorfheide, F. Evaluating dsge model forecasts of comovements. *Journal of Econometrics*, 171(2):152–166, 2012. URL http://ideas.repec.org/a/eee/econom/v171y2012i2p152-166.html.

Hildenbrand, W. On the “law of demand". *Econometrica*, 51(4):997–1019, July 1983. URL http://ideas.repec.org/a/ecm/emetrp/v51y1983i4p997-1019.html.

Janssen, M. C. W. What is this thing called microfoundations? *History of Political Economy*, 23(4):687–712, 1991. URL http://EconPapers.repec.org/RePEc:hop:hopeec:v:23:y:1991:i:4:p:687-712.

Jean-Jacques Herings, P. Universally converging adjustment processes–a unifying approach. *Journal of Mathematical Economics*, 38(3):341–370, November 2002. URL http://ideas.repec.org/a/eee/mateco/v38y2002i3p341-370.html.

Jøsang, A. A logic for uncertain probabilities. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 9(3):279–311, June 2001. ISSN 0218-4885. URL http://dl.acm.org/citation.cfm?id=565980.565981.

Kemp-Benedict, E. General equilibrium as a topological field theory. Quantitative Finance Papers 1209.1705, arXiv.org, Sept. 2012. URL http://ideas.repec.org/p/arx/papers/1209.1705.html.

Kirman, A. Demand theory and general equilibrium: From explanation to introspection, a journey down the wrong road. *History of Political Economy*, 38 (5):246–280, Supplemen 2006. URL http://ideas.repec.org/a/hop/hopeec/v38y2006i5p246-280.html.

Kirman, A. P. Whom or what does the representative individual represent? *Journal of Economic Perspectives*, 6(2):117–36, Spring 1992. URL http://ideas.repec.org/a/aea/jecper/v6y1992i2p117-36.html.

Krusell, P., Smith, A. A., and Jr. Income and wealth heterogeneity in the macroeconomy. *Journal of Political Economy*, 106(5):867–896, October 1998. URL http://ideas.repec.org/a/ucp/jpolec/v106y1998i5p867-896.html.

Le, V. P. M., Minford, P., and Wickens, M. The 'puzzles' methodology: En route to indirect inference? *Economic Modelling*, 27(6):1417–1428, November 2010. URL http://ideas.repec.org/a/eee/ecmode/v27y2010i6p1417-1428.html.

Le, V. P. M., Meenagh, D., Minford, P., and Wickens, M. How much nominal rigidity is there in the us economy? testing a new keynesian dsge model using indirect inference. *Journal of Economic Dynamics and Control*, 35(12):2078–2104, 2011. URL http://ideas.repec.org/a/eee/dyncon/v35y2011i12p2078-2104.html.

Le, V. P. M., Meenagh, D., Minford, P., and Wickens, M. Testing dsge models by indirect inference and other methods: some monte carlo experiments. Cardiff Economics Working Papers E2012/15, Cardiff University, Cardiff Business School, Economics Section, June 2012. URL http://ideas.repec.org/p/cdf/wpaper/2012-15.html.

Lucas, R. E. J. Econometric policy evaluation: A critique. *Carnegie-Rochester Conference Series on Public Policy*, 1(1):19–46, January 1976. URL http://ideas.repec.org/a/eee/crcspp/v1y1976ip19-46.html.

Lucas, R. E. J. *Models of Business Cycles*. Yrjö Jahnsson lectures. Wiley, 1991. ISBN 9780631147916. URL http://books.google.co.uk/books?id=H_x7NAAACAAJ.

Maki, U. Reclaiming relevant realism. *Journal of Economic Methodology*, 7(1):109–125, 2000. doi: 10.1080/135017800362266. URL http://www.tandfonline.com/doi/abs/10.1080/135017800362266.

Mantel, R. R. On the characterization of aggregate excess demand. *Journal of Economic Theory*, 7(3):348–353, March 1974. URL http://ideas.repec.org/a/eee/jetheo/v7y1974i3p348-353.html.

Mas-Colell, A., Whinston, M., and Green, J. *Microeconomic Theory*. Oxford student edition. Oxford University Press, 1995. ISBN 9780195073409. URL http://books.google.co.uk/books?id=KGtegVXqD8wC.

McGovern, S. Dealing with the duhem-quine thesis in financial economics: can causal holism help? *Cambridge Journal of Economics*, 30(1):105–122, 2006. doi: 10.1093/cje/bei049. URL http://cje.oxfordjournals.org/content/30/1/105.abstract.

Mirowski, P. From mandelbrot to chaos in economic theory. 1990.

Mäki, U. 'the methodology of positive economics' (1953) does not give us the methodology of positive economics. *Journal of Economic Methodology*, 10(4):495–505, December 2003.

Musgrave, A. "unreal assumptions" in economic theory: The f-twist untwisted. *Kyklos*, 34(3):377–387, 1981. ISSN 1467-6435.

Penrose, R. and Gardner, M. *The Emperor's New Mind: Concerning Computers, Minds, and the Laws of Physics*. Oxford aperbacks. OUP Oxford, 1999. ISBN 9780192861986. URL http://books.google.co.uk/books?id=oI0grArWHUMC.

Popper, K. *Conjectures and Refutations: The Growth of Scientific Knowledge*. Classics Series. Routledge, 2002. ISBN 9780415285940. URL http://books.google.co.uk/books?id=IENmxiVBaSoC.

Rabin, M. Psychology and economics. *Journal of economic literature*, 36(1):11–46, 1998.

Rabin, M. and Koszegi, B. Mistakes in choice-based welfare analysis. *American Economic Review*, 97(2):477–481, May 2007. URL http://ideas.repec.org/a/aea/aecrev/v97y2007i2p477-481.html.

Rizvi, S. A. T. Responses to arbitrariness in contemporary economics. *History of Political Economy*, 29:273–88, 1997.

Roberts, J. and Sonnenschein, H. On the foundations of the theory of monopolistic competition. *Econometrica*, 45(1):101–13, January 1977. URL http://ideas.repec.org/a/ecm/emetrp/v45y1977i1p101-13.html.

Saari, D. G. Iterative price mechanisms. *Econometrica*, 53(5):1117–31, September 1985. URL http://ideas.repec.org/a/ecm/emetrp/v53y1985i5p1117-31.html.

Sims, C. A. Macroeconomics and reality. *Econometrica*, 48(1):1–48, January 1980. URL http://ideas.repec.org/a/ecm/emetrp/v48y1980i1p1-48.html.

Smets, F. and Wouters, R. Shocks and frictions in us business cycles: A bayesian dsge approach. *American Economic Review*, 97(3):586–606, June 2007. URL http://ideas.repec.org/a/aea/aecrev/v97y2007i3p586-606.html.

Solow, R. A native informant speaks. *Journal of Economic Methodology*, 8(1):111–112, 2002. URL http://ideas.repec.org/a/taf/jecmet/v8y2002i1p111-112.html.

Sonnenschein, H. Market excess demand functions. *Econometrica*, 40(3):549–63, May 1972. URL http://ideas.repec.org/a/ecm/emetrp/v40y1972i3p549-63.html.

Taylor, J. B. Staggered wage setting in a macro model. *American Economic Review*, 69(2):108–13, May 1979. URL http://ideas.repec.org/a/aea/aecrev/v69y1979i2p108-13.html.

Tegmark, M. Importance of quantum decoherence in brain processes. *Phys. Rev. E*, 61:4194–4206, Apr 2000. doi: 10.1103/PhysRevE.61.4194. URL http://link.aps.org/doi/10.1103/PhysRevE.61.4194.

Tieszen, R. Perspectives on intuitionism. *Philosophia Mathematica*, 6(2):129–130, 1998. doi: 10.1093/philmat/6.2.129. URL http://philmat.oxfordjournals.org/content/6/2/129.short.

Weintraub, E. R. *Microfoundations: the compatibility of microeconomics and macroeconomics*. Cambridge University Press, 1979.

Wickens, M. R. How useful are dsge macroeconomic models for forecasting? CEPR Discussion Papers 9049, C.E.P.R. Discussion Papers, July 2012. URL http://ideas.repec.org/p/cpr/ceprdp/9049.html.

Zadeh, L. Fuzzy sets. *Information and Control*, 8(3):338 – 353, 1965. ISSN 0019-9958. doi: http://dx.doi.org/10.1016/S0019-9958(65)90241-X. URL http://www.sciencedirect.com/science/article/pii/S001999586590241X.

Chapter 3

Macroeconomic Fluctuations from Hierarchical Enforcement Avalanches

3.1 Introduction

The death of the Representative Agent is not the end of Macroeconomics. In the previous chapter I gave compelling reasons why macroeconomics based on General Equilibrium can not give us the whole dynamic picture of economic aggregates, but this should only open our eyes to the myriad of possible alternatives. In this chapter I attempt to lay foundations for a new path to modelling macroeconomic volatility — what might be inaccurately described as the *Business Cycle*. The non-equilibrium model I present instead reproduces this aggregate volatility of boom and bust, based on a hierarchy of enforcement relationships which occasionally undergoes a cascade of failing enforcers that prevents productive agreements being fulfilled.

The chapter is organised as follows: section 3.2 reviews the motivation for the enforcement perspective, and seminal philosophical treatments of third party enforcement; section 3.4 gives both intuitive and technical explanations of the model; while section 3.5 describes the method used to assess this model as a description of macroeconomic data and 3.6 gives the outcome; finally, section 3.7 provides a summary of my findings, a proposed extension to the other great question of economics, and bottom up policy implications.

3.2 Enforcement and hierarchies

3.2.1 Why Enforcement?

A tempting response to the criticisms of the previous chapter would be to try and build some other model of exchange, using other well established tools from microeconomics like Game Theory. After all, it would be far less painful to overcome the unrealistic assumptions of the Walrasian Auctioneer and single price market by a more realistic representation of how markets operate, while still retaining familiar ideas from decision theory. There are two important disconnections between microeconomic theory and observed reality that limit its intuitive relevance to building useful aggregate models: first, the irregularity of human choice when compared to decision theoretic descriptions; second, the irregularity of our myriad modes of exchange, once we stop pretending that multilateral single price markets describe it all, and the inevitable interaction of that exchange with other aspects of society.

Game theory gives a useful language for discussing decisions when facing regular scenarios, by giving a benchmark in which there is a sense of optimality towards which decisions strive. However, in practice this description does not offer good predictions of individuals' behaviour in strategic situations (Binmore, 2007; Rabin, 1998). Game Theory also suffers from the problem that Nash equilibria, where no individual has an incentive to change their behaviour, often isn't specific enough to give a single clear prediction. All this said, crucially, no other decision theoretic description seems to perform much better (Harless and Camerer, 1994). A more definite limitation of Decision Theory generally is seen in those decision situations that involve unknowns that cannot be described by probability distributions, because even the range of outcomes isn't known (Binmore, 2007). This is what is known as *Knightian Uncertainty*, and I discuss it in more detail in the previous chapter. It is important to realise that this uncertainty is different from that treated in Bayesian Game Theory, which revolves around the insight of Harsanyi that equilibrium beliefs must promote a behaviour that is a best response

to other players' best response, given their own equilibrium beliefs. That it actually pays the agents to behave in this way assumes one of two things: if they are rationally choosing their response, that they act as if there is a known world bounding future possibilities, to which they can respond rationally; if they are adapting their beliefs and behaviours over time, that then the environment must be stable enough to allow an equilibrium to be found. For both, the decision has to be in a regular setting, in which pressures act on beliefs much more quickly than the regularities of the environment that define the decision change. This is almost certainly not the case in any system that we can not yet describe, like technological advance. So any decision problem that is likely to be subject to faster technological change than learning is not a good subject for game theory. Clearly if we are considering all the decisions taking place across the economy, then unknowable technological change will have a significant role.

Even if we could describe the parameters of every decision being made in the economy, we would still have the problem that there is no obvious way to aggregate all these specific decisions, or which could be abstracted away. Indeed, as Kirman (2010) explains, just describing exchange would involve describing a vast array of different institutional arrangements, be they multilateral, bilateral, or otherwise. Aggregating this variety in a feasible way would have to involve finding more generalities between them than has thus far been found. The solution of Neoclassical Economics was to assume a Walrasian Auctioneer that could not actually represent a real process, as I explained in the previous chapter. The Auctioneer's lack of realism was perhaps an inevitable consequence of Economics's focus on prices; it seems to me that there couldn't be a worse focus than prices for a general description of exchange, because the way they are determined is one of the most obvious differences between different kinds of exchange — for instance consider the varied mechanisms described in Auction Theory. That said, hoping to find regularity across all exchange seems optimistic when one considers that the vast amount of exchange that goes on depends on the differences between the objects being exchanged.

This incompatibility, between microeconomics and the generalisations needed to simply aggregate the great variety of economic activity, suggests that a more fruitful kind of Reductionism in Economics might revolve around some other person level behaviour besides choice and exchange. Of course, once we stop focussing on markets we see a wider problem with economic theory: the assumption that economic exchange is somehow orthogonal to other social interaction is the only way that a picture involving only markets could be supported. However, there are well known examples of social unrest and change following economic depression. In fact macroeconomic time series share similar properties with series describing civil conflict, which I will describe in more detail in section 3.4.

Rather than address all these problems head on, then, with some even more elaborate descriptions of human decision making, I abstract from exchange and focus on the enforcement that mediates it — and much of the rest of social interaction. My justification is that we can describe general features of enforcement quite simply, because unlike exchange it doesn't depend on the differences between the objects we are describing. I argue that the general features we can describe are enough to give us aggregate volatility. This would mean that we could abstract away from much of the activity around enforcement, without compromising the realism of the relationship between our description and macroscopic society.

There are, of course many modes of enforcement, but I feel only one puts one component in a position to impact a significant portion of the rest of society: third party enforcement, where agreements between two individuals are enforced by the threat of punitive action from a third party with asymmetric power compared to the individuals. This influence of the one enforcer on the aggregate is important to produce aggregate dynamics, because if the aggregate uncertainty is going to come from the components, then it must come from the natural uncertainties associated with the behaviour of each component. But, if all the components have equally small influence over the others, then the Law of Large Numbers tells us that the uncertainties around each will can-

cel out over the whole of society. Of course the dynamics of the aggregate could be driven by things on the scale of the aggregate, but then we are left with the problem of identifying these sources of uncertainty.

It is precisely because of the Law of Large Numbers that, in the neoclassical story, fluctuations in the aggregate economy are not the sum of separate fluctuations in its component markets. Instead, the aggregate dynamics do have to be driven by some exogenous force at the aggregate level. What that exogenous force represents is hard to pin down. In the previous chapter I mention some complaints, along these lines, raised against the prominent New Keynesian DSGE model of Smets and Wouters. More generally, however, "technology" shocks are attributed the greatest effect, and yet are often explicitly regarded as unobserved. If we were to consider them as literally technology shocks, then we might struggle to identify a technological change that simultaneously affected enough of the economy to have a decisive impact. After all, technological adoption is not instantaneous, and even something as world changing as the internet took a decade to gradually integrate into the way work is done over the whole economy. A common alternative is to interpret the technology shocks more broadly, so as to include energy supply. Finn (1995) found the Solow residual to have a correlation of -0.55 with Oil prices, which could be taken for a connection. However, the price of Oil is almost certainly endogenous and the causal direction with productivity reversed, and this is what Finn concludes from her analysis.

Recent work by Acemoglu et al. (2012) illustrates how the right interrelationship between market members could give rise to significant aggregate fluctuations, arising from shocks to only those members. Integrated with market equilibrium this model has all the problems described above. It also has no explicit dynamics, and the network structure is exogenous with no reason given for why the economy should remain in such a volatile state. Nevertheless, the asymmetry they describe in actors' influences seems a fair means for local failures to propagate upwards to the whole economy, and it is in this spirit that I look to the asymmetric relationship between enforcer and en-

forcees. The model I construct here will be dynamic, and will justify the network structure endogenously. But, it cannot be assessed empirically in the same way that Acemoglu et al. (2012) illustarte their model. This is because that model relates the network structure to only the relationships for the exchange of factors of production, while the model here will involve a network of enforcement relationships.

3.2.2 Existing Descriptions of Enforcement

My conclusions from the previous chapter were that Science should start its search for new descriptions of phenomena with language that has been useful for describing other phenomena. Along these lines, my choice to focus on enforcement is motivated by the very long tradition in renaissance thought of describing third party enforcement, along with the hierarchies of third party enforcers that I will soon expand on.

The modern liberal tradition on third party enforcement was initiated by Hobbes (1651), with his Sovereign as the only thing standing between social order and anarchy:

> The finall Cause, End, or Designe of men, (who naturally love Liberty, and Dominion over others,) in the introduction of that restraint upon themselves, (in which wee see them live in Common-wealths,) is the foresight of their own preservation, and of a more contented life thereby; that is to say, of getting themselves out from that miserable condition of Warre, which is necessarily consequent (as hath been shewn) to the naturall Passions of men, when there is no visible Power to keep them in awe, and tye them by feare of punishment to the performance of their Covenants...

This interpretation of subjugation as self-initiated might grate against the ideals of Western Liberalism, but is intuitive enough when we consider the day to day instances in which we willingly defer authority to others. These day to day examples of authorities within smaller social groups also points to the natural extension of Hobbes'

description, to different levels of what we might call a social hierarchy. As economists we are perhaps used to ignoring the importance of governments' enforcement, but Dasgupta (2003) is quick to point out its indispensability to the market's operation: it might become invisible through habit, but the legal system implicitly mediates between firms, banks and grocers simply by allowing the possibility of punitive action. It guarantees the quality of goods, and the payment for them – whatever its medium. It restricts the practices of manufacturers, to minimise negative externalities, and it ensures that investors can fuel innovation with the safety net of bankruptcy. The trust that lies behind all these anonymous market transactions, then, is ultimately backed by the overwhelming martial authority of the state.

Once invested with authority, however, one might question what incentive there is for the sovereign to actually carry out their enforcement role. This raises the question of why people should continue to trust the institution of third party enforcement? Hobbes (1651) would respond that out of want for taxable income, or if nothing else then for paternal vanity, the sovereign's own self-interest would drive it to nurture a wealthy efficient society. Meanwhile, Locke (1821) would argue that a "tyrannical" sovereign's subjects would simply unseat it and select a new government. It is Locke's mechanism that Dasgupta chooses, arguing that a democratic government under the scrutiny of a "free and inquisitive" press has incentive enough not to err from its sovereign duties. Thus, he suggests citizens of such a government should trust the legal framework. Further, with the Arab Spring fresh in our collective memory, we might be tempted to believe that the machinery of democracy is hardly a prerequisite of an effective *right to revolution*. It can easily be argued, then, that the effectiveness of third party enforcement depends on the fact that enforcers are removed from their role when they fail in its execution. Of course enforcers may fail in their role for other reasons, and it's also reasonable and intuitive to believe that sometimes enforcers are removed not because of malfeasance, but because they have been unexpectedly left without the resources they need to carry out their role. I will rely on this argument for the cause of collapses in the model, and hence the origin of aggregate volatility.

One of my arguments for this choice of component to a macroeconomic model, over the more traditional decision theoretic microfoundations, was that they are more common than centralised markets — i.e. they approximate far more of the local behaviour of the people that make up an economy, or indeed society. To convince the reader of this claim, let us consider some commonplace examples of behaviour that can be effectively described in terms of hierarchical enforcement, with the possibility of collapse:

The Firm A firm's production is contingent on the coordination of efforts by a large number of employees. Should employees cease to cooperate, because one is trying to do less than their share of the work, then a manager exists to monitor and mediate. In this way, the manager is acting as a third party enforcer, because they can administer punitive action against those not cooperating for the greater good. We might also note that in large firms managers tend themselves to have managers, arranged in a hierarchical structure. Collapse is seen whenever employees lose faith in their manager and cease to carry out their roles as effectively, an presumably familiar occurrence for anyone who has worked in a sufficient number of organisations.

The Union A union's power to negotiate with wage setters is entirely contingent on the members' coordinated refusal to work, as an inherent agreement between them. Monitoring this coordination and keeping track of the scabs is done by the head of a local chapter or branch, who is acting as a third party enforcer. Again, these heads are themselves monitored by third party enforcers further up the structure, who can also take punitive action against malfeasant heads, and chapters that betray the other chapters. Collapses can take place when the members of a chapter put a vote of no confidence in their representative.

The Black Market The defining characteristic of the Black Market is that it foregoes legal enforcement of contracts. Many agreements will instead rely on the value of protecting ongoing relationships as collateral for agreements, and I begin to

explore this mechanism in chapter 5. Still, we are all familiar through popular culture, and some through more direct experience or anecdote, of organised crime with a leadership structure that allows agreements to be mediated by a third party *capo* or *boss*. Collapses here are certainly part of the common story, and tend to be a lot more bloody!

Law Enforcement Both branches, of enforcement and justice, in the UK have tiered structures. The third party enforcement afforded by these structures helps prevent all parties from becoming corrupted and undermining the function of the whole system.

Bank Runs Although they also carry out the functions of matching and liquidising, banks too can be thought of as third party enforcers between creditors and debtors: as punitive action they deny further membership in the system of credit and lending to those who do not meet their agreements. In 2008 and previous recessions we saw the spontaneous collapse of some enforcers in this system, and the subsequent flight from savings that severely reduced lender-borrower agreements.

Feudalism The most easily recognised social hierarchies are those enshrined in the institutions of feudal societies. An example of contemporary feudalism, where our observations can't be subject to historical revisionism, is that of Saudi Arabia. In Saudi Society it is a central role of the members of each level of the hierarchy to adjudicate on disputes between members of the level below.

3.3 Network notation and existing models

Because the hierarchiesof section 3.2 are elaborate networks of changing relationships, I will describe them formally using Graph Theory and Network Science. The mathematics of graphs and networks is now fairly mature, and there is relative uniformity

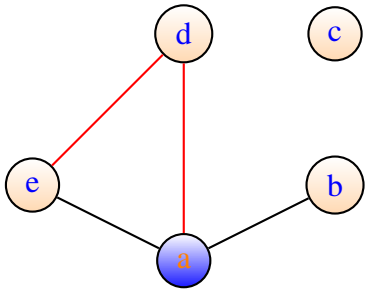


Figure 3.1: Example network

of terminology and notations. In order to summarise the more substantive theory, I'll first, here, list the necessary definitions of terms used to describe networks. As Jackson (2010) has provided such well rounded summary text, I will follow his notation where there is ambiguity, and then adapt others' to conform later:

Node Those objects which are linked within a network —this could be an agent, a firm, or some more abstract object in some uses: $i \in N$. For example a in figure 3.1 is a node, as are b, c, etc.

Edge; Link A connection between two nodes, denoting a relationship of some kind between them: $ij \in g \subseteq \{(k,l) : l,k \in N\}$. For example, the line connecting a, and b, in figure 3.1 represents an edge.

Graph; Network A collection of Nodes and Edges: (N, g). The entirety of figure 3.1 is an example of a graph.

Neighbourhood (immediate) The set of nodes which share an edge from g with a particular subject node, i: $N_i = \{j \in N : ij \in g\}$. In figure 3.1, the neighbourhood of a would be $\{b, d, e\}$.

Degree The number of other nodes to which a particular subject node, i, is connected; the cardinality of that node's immediate neighbourhood: $d_i = \mid N_i \mid$. For instance, the degree of a is 3.

Neighbour A node which shares an edge from g with a particular subject node, i: $j \in N_i$. So b is just one neighbour of a.

Neighbourhood The set of nodes that can be reached by crossing no more than a certain number, d , of edges of g, $i : N_i^d = \{j \in N : \exists\, p = (ii_2, ...i_{I-1}j)\ , |p| \leq d\}$

Path A sequence of edges between two nodes, i and j say, including other nodes at most once: $(i_1 i_2, ...i_{I-1} i_I)$ s.t. $i_k i_{k+1} \in g \quad i_1 = i,\ i_I = j$ and$i_k \neq i_l \forall k, l \in N$. For instance, (ad, de) is the path shown in red in figure 3.1.

Connectedness Two nodes are connected when there is a path between them. A graph is then called *connected*, if every pair of nodes is connected, and *disconnected* otherwise.

Component A subset of nodes between every pair of which there is a path, but from no element of which there is a path to any other node: $\bigcup_{n=1}^{\infty} N_i^n(g)$. So, in figure 3.1, the set $\{a, b, d, e\}$ is one component, while $\{c\}$ is the other.

Diameter The largest, among all pairs of nodes, of shortest paths between them. Where a graph is disconnected, this often ignores all nodes not in the largest component. So in figure 3.1 the diameter is 2, because two edges have to be travelled to get from b to e, while all other pairs of nodes, in the largest component, can be traversed using a single edge.

Tree A network or component in which there is only one path between any two nodes.

Star network A tree in which one node shares an edge with every other node; a hub and spokes formation.

Complete Graph A network in which every possible link is present.

Though the distribution of nodes' degrees alone captures useful features of a network, it by no means describes everything on which an agent network's dynamics might depend. As such I now give some popular measures of other network properties:

Clustering Clustering roughly captures the probability with which neighbours share a common neighbour, a kind of density of the network's edges. Within this loose definition there are several more specific measures employed in practice. The most common measure of clustering is the relative number of completed triads, that is, the number of occasions when two neighbours are also neighbours:

$$Cl(g) = \frac{\sum_{ij,ik \in g} \mathbf{1}(jk \in g)}{|ij, ik \in g|}$$

Recent theoretical work suggests that a subtly different measure of density is in fact more relevant to contract enforcement (Jackson et al., 2012). This *support* is defined as the number of edges whose corresponding nodes share a common neighbour:

$$S(g) = \frac{\sum_{ij \in g} \mathbf{1}(\exists k \in N \text{s.t.} ik, jk \in g)}{|g|}$$

The only difference is, of course, the weight given to each triad. However, the second measure is argued to better capture the way in which the presence of triads contributes to peer-pressure-like enforcement; I will expand on this point below, where I discuss dynamic models of Social Capital.

Centrality In contrast to clustering, which is a global measure of the localisation of relationships, *centrality* describes the directness of a specific node's connection to all other nodes in its component. Where indirect connections count for nodes' utility, centrality gives an idea of the importance of a particular node to the efficiency of a whole network. The crudest measure of centrality is *degree centrality*, the proportion of other nodes to which a node is connected. Degree centrality, however, fails to account for indirect connections, and can therefore easily miss the property we want to capture. A richer measure is the *betweeness centrality*, which captures the frequency with which a node lies on the shortest paths between other nodes:

$$Ce_i^B(g) = \sum_{\neq k \neq j : \neq k \neq i \neq j} \frac{\text{proportion of shortest paths } j \to k \text{ including i}}{(n-1)(n-2)/2}$$

A more abstract concept of centrality has recurrently been found to be important in the theoretical literature. This so called Bonacich centrality is defined recursively from the matrix representation of a network, wherein the presence of edges, $ij \in g$, is expressed as the presence of a 1 in the ij^{th} entry of a matrix (also denoted g, for convenience). This representation is useful because indirect connection, of n links, are captured by the n^{th} power of the matrix. With this notation in hand, one can define Bonacich centrality as proportional to the sum of neighbours' Bonacich centrality:

$$C^e(g) = \frac{gC^e(g)}{\lambda}$$

Intuitively, this captures the idea that a node becomes more important to the structure of a network, the more often they act as a bridge between other important nodes. Using matrix notation allows the well understood techniques of linear algebra to be brought to bear, and the above definition clearly makes Bonacich centrality an eigenvector[1] of g. Intuitively, if we think of pre-multiplying a vector of quantities by g as investing each quantity, of a perishable good, by the node of the same index in their available neighbours' production. As an eigenvector, the Bonacich centrality is an initial allocation of the perishable good, for which everyone investing the same particular proportion of their stock in each neighbour's (one to one) production leads to the same allocation next period - that is, larger stocks invested among several less central nodes with smaller stocks are compensated exactly by there being more neighbours of that centrality than those neighbours themselves have[2].

[1] That is, the basis vector of a dimension which is mapped to itself by g.

[2] This may still sound counterintuitive, as fixed proportions of the higher centrality going out will be larger than the same proportion of their less central neighbours' stocks coming back along the links. It is the fact that g is conventionally given ones along its leading diagonal that balances the equation; that is, the nodes invest the same proportion of the perishable stock in their own production. So a bigger stock leads to more stock produced locally next period

Power Law and other fat tailed distributions

Power law distributions seem to describe at least the upper tail of the distributions for many key economic variables (Kirman, 2010; Gatti et al., 2008), and indeed all *stable* distributions[3] besides the normal look like power laws in their upper tail. These other stable distributions' infinite variance has unpleasant implications for many statistical techniques, and for standard economic intuition. Nevertheless, to ignore fat tails despite their ubiquity would be to ignore their potentially deep structural implications. Indeed, leptokurtosis of macro-errors need not suggest infinite variance of micro-errors; if the superposition principle breaks down because the micro-errors are not independent, then summation is not the appropriate description of aggregation and the stable distributions will no longer be attractors. With structural correlation in the system, finite variance micro-errors might still give rise to macro-kurtosis through cascades (Bak et al., 1993).

Power Law distributions are characterised by the probability mass function, or probability density function:

$$f(x) = ax^{\gamma}\, a,$$

This seemingly over simple form gives them the property of being scale-invariant, which is to say that the relative probability of different values depends only on their relative size, rather than their absolute size. This then implies that the scale against which they are measured does not matter.

[3] That is, distributions which are attractors under summation and normalisation. This translates as those distributions that would tend to arise in systems of superpositioned objects over multiple scales. The best known stable distributions are the Levy, Cauchy, and Normal (for which the vanilla Central Limit Theorem shows stability) distributions.

3.3.1 Static network formation

Economics, particularly macroeconomics, has become increasingly focussed on equilibria derived through fixed point proof (Giocoli, 2003). This means a science that has very well developed statics, but whose orthodoxy pays little heed to the dynamic behaviour of objects (Dopfer, 2011). This is an obvious drawback in prediction of an inherently dynamic world, but so long as those statics can be shown to approximate the limit of some dynamic model[4] they may well be relevant. Indeed, I would still imagine these models to be predictively useful, when the convergence of a system to equilibrium is fast enough compared to the arrival of disturbances[5]: in which case the equilibria are dots which can be joined for an approximate trajectory of the system. For the sake of integration with this orthodoxy, and the discipline's existing human capital, I will begin my summary of network modelling with those frameworks that focus exclusively on statics.

The stylised facts of static networks

As discussed first in chapter 1, interest in networks stems from a concern that social systems are too non-linear for their behaviour to be usefully approximated by that of individual components. In order for a system to be otherwise, the behaviour and properties of any individual component must be as good as independent from its neighbours'. It is a general departure from this independence, in the presence of relationships, that is the domain of the networks literature —indeed, heterogeneity of structure is a possible explanation for the distribution of other heterogeneous properties across the popula-

[4] As is indeed suggested by the experimental research of Binmore (1998) and others; these experiments suggest that decision making adapts from some predetermined broad heuristic towards a rational response to others' behaviour and the environment, but that this may take some time depending on the complexity and persistence of the game, and the power of its incentives. Indeed, Grosskopf and Nagel (2008) more recently find evidence that further supports such adaptive behaviour over the second alternative of rational response to irrationality - which is often observationally equivalent.

[5] This is contrary to the more dogmatic position of some Complexity oriented modellers, who seem to prefer a realistic treatment of time even when it is at the expense of valuable mathematical precision (Kirman, 2010; Gatti et al., 2011)

tion, as I will soon discuss. In order to see this heterogeneity, we need to contrast real networks with the benchmark structure of Gilbert's random graph model, in which links are formed independently of neighbouring links.

Formally, the Gilbert model has every edge, between a set of nodes N, be present with equal probability:

$$P(ij \in g) = p \,\forall\, i, j \in N$$

As this framework leaves each node with a degree from the binomial distribution, which is approximated by the (more combinatorially manageable) Poisson distribution, such graphs are also dubbed Poisson Random Networks.

It turns out that this random attachment does not adequately describe the distributions of degrees generally observed. These are better described by power law distributions, or some weighted average of a power law and a Poisson (Jackson, 2010; Kirman, 2010). That is to say that there are more nodes with high degree, and more nodes with low degree, than one might expect of random link formation. As such, it is argued that some *preferential attachment* must play a role in the formation of those networks, whereby the probability of a link being present is affected by those links already present around it.

Power laws are not just common among degree distributions, but are actually present everywhere in nature, and particularly in social science data. Vilfriedo Pareto was the first to recognise such a power law distribution, and the associated scaling behaviour, in data. He originally noticed that income distributions tended toward a power law in their upper tail, and thereby earned such distributions the name Pareto. This result has been confirmed many times since. More recently the distributions of both US and international firms' sizes have been found, by several authors, to follow power laws. Less well studied, the productivity distribution of firms has been shown to be fat tailed (Gatti et al., 2008). More strikingly, stock market returns have been suggested to follow

a power law, so that crashes needn't be dismissed as outliers (Gabaix, 2009). The prevalence of fat tailed distributions in macroscopic data could be taken as evidence of interrelation among micro-components.

A long standing, and popular, feature of observed networks is the small worlds phenomenon famously highlighted by the six degrees of separation experiment. Simply put, real networks seem to have much smaller diameters than random networks, so that any two nodes are joined by far fewer links than one might expect. That this result is counter-intuitive lends credibility to random networks as a benchmark. Despite this small diameter, however, the small worlds phenomenon is generally seen to coexist with higher levels of clustering in social networks, than one would see with random connections (Jackson, 2005). That is, even though there tends to be a pretty direct path between any two agents, most agents' neighbours are very likely to be connected to exactly the same agents.

As well as there often being more nodes of high and low degree observed than one might expect, there is at least anecdotal evidence that these extreme nodes are themselves related by more than uniform probability. This correlation of neighbours' degrees is termed *positive assortativity*, and is hypothesised to routinely take different signs in broadly different contexts (Jackson, 2010). Jackson gives the example of a negative correlation in the network of trade agreements between countries, with the hypothetical explanation that powerful countries tend to attract collections of weaker cronies.

Random networks

Network models that randomly assign links to pairs of individuals, such as the Gilbert model, are quite obviously not the product of any explicit interactions between components. Instead, they could be regarded as holistic attempts to reproduce the static stylised facts of networks, without any reductionist extrapolation from the behaviour of their components —and hence less robust when carried over to previously unob-

served domains of the variables, as discussed in chapter 2. Though this makes the structure exogenous, it still allows the modelling of components' behaviour within the particular network, and the prediction of aggregate variables besides those describing network structure. For these predictions to be useful, we would have to assume that any effect on network structure, of the endogenous variables' behaviour, was negligible. There are reasonably tractable random static graphs that capture some of the stylised facts detailed in the previous section.

My interest is in social engineering, which relies on social networks having endogenous structure. Nevertheless, there may be aspects of the relationships between individuals that may change too slowly to be endogenous, such as the physical locations of population centres or resources, that are best described by a random graph. I will therefore briefly elaborate on Random Networks here.

Perhaps the simplest elaboration on Gilbert's Poisson Random Graph is the Markov Random Graph. Here, the principal graph is obviously still stochastic, but the distribution of links on this graph is parametrised by a second, constant, graph. This second graph has all the potential links of the first graph as its nodes, while the presence of edges between these nodes signifies inter-dependence between the probabilities of those links appearing in the random graph. This means that any independently formed links, in the first graph, are represented in the second graph by nodes with no connections. The advantage of such a description, of a non-independent graph's link distribution, is that it gives a neat expression for the probability of the entire graph. Unfortunately, interesting properties, such as the degree distribution, are not so tractable (Jackson, 2010).

Degree distributions are much more manageable when using the Configuration Model and Expected Degree Model. This is simply because both pre-specify a degree distribution which then acts as a constraint on the random network formed. Such graphs only behave definitively like social networks in the limit, and so they may only be used to prove limiting results for given network properties. Of course a limiting behaviour

is not necessarily usefully close to the behaviour of a particular finite population. Jackson suggests mixed inference from limiting results on general parameter values, and simulation from specific parameter values. This does perhaps make for a good guard against sensitivity to parameter values, but it is the simulations here that do the bulk of the work. I will not, therefore, go into any more detail on these two approaches, though a full explanation can be found in Jackson (2010).

Game theoretical networks

Holistic reproductions of networks are perfectly scientific but, as in Lucas' famous critique (**?**), we should be wary of assuming that some object's behaviour will be consistent out of sample (whether in terms of time or space) when the components of that object may vary in their behaviour or numbers. For this reason we might seek to break away from networks that relegate components' behaviour to uncertainty, and instead let the structure emerge endogenously from explicit behaviour of the components. For static models we must expect all forces of individual behaviour to balance out. That is, we expect the system of components, that make up the whole, to be in equilibrium.

If we are explicit about the origin of the forces acting on the components, and we believe that human decision making tends toward rational behaviour, then the equilibria of the system will be described by the pure or mixed strategy Nash equilibria of some strategic game.

A *pure strategy equilibrium*, $\underline{s} \in S$, ascribes to each individual a specific *strategy* — a set of contingent actions — from the set of all such combinations, S. The equilibrium derives from that fact that no individual would gain from deviating, so long as every other player subscribes to their corresponding strategy:

$$u_i\left(s_i, \underline{s}_{-i}\right) \geq u_i\left(s, \underline{s}_{-i}\right) \forall s \in S$$

Meanwhile, a *mixed strategy equilibrium* describes a tuple of subjective probability

distributions, $\underline{\sigma} \in \Delta^{rn}$, for the r strategies available to each of the n players. When a player describes their uncertainty over their opponents' actions by these distributions, it is optimal for them to play their own strategies with the probabilities described by their own distribution — though what this randomisation could mean in terms of actual human decision making is unclear.

$$Eu_i\left(\sigma_i, \underline{\sigma}_{-i}\right) \geq Eu_i\left(\sigma, \underline{\sigma}_{-i}\right) \forall \sigma \in \Delta^1$$

An early model of strategic network formation is that of Aumann and Myerson, which orders the possible links in the network, and then sequentially allows the nodes joined by those links to choose whether or not it should exist. When a link is chosen to be omitted the next link in the sequence comes under consideration. When a link is chosen to be in the network, the process returns to the first potential link in the sequence that was already omitted. There is then another chance for the two nodes to allow that link. Once allowed however, a link can not subsequently be omitted from the network. Despite this *on second thought* feature, ordering significantly affects the equilibrium networks formed. Moreover, the game is typically one of full information, where all chosen links and all players' future behaviour figures in the decision calculus. It is therefore very difficult to analyse for any but the most simplistic networks, and no general results have been developed (Jackson, 2003; Demange and Wooders, 2005).

A more tractable network formation game has all players simultaneously pick their desired neighbours, and then those mutual selections confirmed as edges of the graph. Though this contemporaneous formation makes analysis more straightforward, there are too many equilibria, of too great a variety, for anything useful to be said. It is for this reason that noncooperative game theory is often given up in favour of refined equilibrium concepts, specific to network formation (Demange and Wooders, 2005).

Matthew Jackson (2003) and various co authors are responsible for several network specific equilibrium concepts. The earliest of these was *pairwise stability*, but it still

finds a place in Jackson (2010). For a network to be pairwise stable, two conditions must be satisfied:

$$\text{PS1:} \quad \forall ij \in g \; u_i(g) \geq u_i(g - ij) \text{ and } u_j(g) \geq u_j(g - ij)$$

that is, no party to a link should have greater utility if that link is removed from the network; and,

$$\text{PS2:} \quad \forall ij \notin g \text{ if} u_i(g + ij) \geq u_i(g) \text{ then } u_j(g + ij) < u_j(g)$$

that is, if some party to an absent link can increase their utility by introducing that link, then the other party must lose utility by its introduction. Together these conditions ensure that when two agents both gain from having a link, all else being equal, then it will exist. This is in contrast to the Nash concept, where it is also an equilibrium for the link to not exist — there being no gain to requesting the link should the other party not. Jackson justifies his departure from a noncooperative framework as a simplifying assumption to achieve obvious conditions.

A further step towards cooperative game theory is taken with the concept of *strong stability* (Jackson, 2003). Here, coalitions of more than two nodes are considered, so that more than one link is under consideration at any one time. Explicitly, strong stability requires that there be no Pareto improvements for members of some set of nodes S, whenever that *coalition* of nodes can *obtain* another network, g', via the following conditions:

$$\text{SS1:} \qquad ij \in g' \text{ and } ij \notin g \Rightarrow \{i, j\} \subseteq S$$

that is, the coalition, S, can unilaterally add any link needed to obtain g' from g because both parties to every additional link needed belong to the coalition; and,

$$\text{SS2:} \quad ij \in g \text{ and } ij \notin g' \Rightarrow \{i, j\} \cap S \neq \emptyset$$

that is, the coalition can unilaterally delete any links to obtain g' from g because at least one party to every excess link belongs to the coalition. So when both these conditions hold, and g' is *obtainable* from g, then it cannot Pareto improve the lot of S:

$$\text{SS3:} \quad \text{SS1 and SS2} \Rightarrow \forall i \in S \text{ if } u_i(g') \geq u_i(g) \text{ then } \exists j \in S \text{ s.t. } u_j(g') < u_j(g)$$

Though this is a fairly natural extension of pairwise stability's abstraction from negotiations, it suffers the eternal problem of cooperative game theory: once network formation begins, rational agents would need some credible threat to stop them defecting from a coalition. In the absence of this threat, the agents would therefore have no reason to fear any other coalition. Modellers of other networks might be happy to ignore such details, but as I am interested specifically in the enforcement of agreements it would be inconsistent to adopt a framework with such a hole.

3.3.2 Dynamic network formation

Where the tendency of a system to equilibrium is much slower than the rate of disturbances to that system[6], the equilibria cannot be hoped to usefully approximate future behaviour. They may even be irrelevant. To guard against this possibility, Evolutionary Economics models economies composed of individuals whose behaviour deviates *systematically* from that which would be rational[7]. Instead, individuals' strategies are based on an induced model of the world, which adapts with each new piece of information. It is those successful strategies that are most likely to persist, and so the

[6] As could fairly be supposed in a world of rapid technological change, globe trotting diseases, and chaotic climate.

[7] Which is to say, even the mean behaviour of agents does not correspond to the rational behaviour.

behaviour is seen to evolve. This is not inconsistent with perfectly rational equilibria, as these have often been shown to be limits of such adaptive processes (Binmore, 1998).

Mean field approximations and Representative Agents

A familiar difficulty of dealing with systems of interacting heterogeneous agents is its mathematical intractability. No small part of that is contributed by the combinatorial problems of simply counting how many of each type of relationship are present. This problem is often countered by the technique of *mean field approximation*, which replaces many bodies and their interactions with a single body and a representative external *field*, capturing the effects of all the interactions. If this sounds a lot like Representative Agent analysis, that's because the RA is an example of a mean field approximation; so all the criticisms given in chapter 2 should counsel caution in using such an approach, especially dependence on the law of large numbers (Gatti et al., 2011). Still, while the RA assumes a star shaped network with the Walrasian auctioneer at the centre, mean field approximations can be taken for networks with more nuanced and localised interaction than in the DSGE case.

Because of its conventional description in the Hamiltonian notation of physics, and a lack of such uniform notation for many body economic systems, mean field approximation is difficult to formalise in a general framework. Informally, it involves:

1. Representing agents' characteristics as their stochastic deviation from mean;
2. Expanding the description of their dynamics and effects on one another in terms of means and deviations;
3. Discarding higher order terms.

When and where mean field approximations are good approximations remains largely unknown, and is generally explored case by case using simulations (Jackson, 2010).

Dynamic stylised facts

As a relatively new area of interest, with inherent difficulties and additional costs in its measurement, Network Science does not yet have any canonical stylised facts for co-movements of network measures. Nevertheless, there are dynamic stylised facts regarding the distribution of the network's population, for at least economic variables. For instance, Gatti et al. (2008), find that the size distribution of firms varies systematically over the business cycle, but that this variation follows no discernable pattern between countries.

Dynamic Random Networks

The obvious random dynamic network is a variation on the Poisson static random graph, described in the previous section, which grows over time as new nodes are added. Each new node randomly attaches to a fixed number of existing nodes, m, and in this way degree distributions of old nodes grow up from m links with time. Approximating this process with a continuous one and then using a simple mean field approximation, it can be shown that the degree distribution after a large number of periods is approximately the exponential distribution,

$$F_t(d) = 1 - e^{-\frac{d-m}{m}}$$

Clearly an exponential distribution does not have the fat tails observed in most real networks, and so some form of *preferential attachment* must be introduced — as it was with Markov random networks, or the configuration model in the previous section. The simplest way to produce more high degree and low degree nodes is for the m initial attachments to be made with more probability to high degree nodes. Jackson (2010) simply makes the probabilities proportional to the existing node's degree. Using the same two approximations, we find that the distribution function of degrees is now a power law with exponent -3,

$$F_t(d) = 1 - \frac{m^2}{d^2}$$

Though they are fat tailed, degree distributions tend to lie somewhere between power laws and random distributions. In light of this fact the above models can be combined, with some proportion, α, of new links assigned randomly and the remainder with preferential attachment. This scheme leads to an approximate cumulative distribution of,

$$F_t(d) = 1 - \frac{m + \frac{2\alpha m}{1- \ alpha}}{d + \frac{2\alpha m}{1- \ alpha}}^{\frac{2}{1-\alpha}}$$

Despite their dynamic nature, growing random graphs are most important for their long run structural features, rather than necessarily reproducing dynamic behaviour of real networks - though of course the reason for their desirable properties might well be related to an accurate imitation of real network growth. As well as a fat tailed degree distribution, they can also show positive assortativity. The clustering seen in real networks is, however, not present in the limit. To counter this, the preferential attachment of the previous models can be modified so that rather than having increased probability of attaching to any high degree node, new nodes attach first uniformly at random to m_r nodes, and then uniformly to m_n neighbours of those m_r initial links (Jackson and Rogers, 2005a). In this way high degree nodes are favoured (as these are more likely to themselves be neighbours of the random initial neighbours) but additional probability is also given to sharing neighbours, that is, to clustering. This leads to a fat tailed degree distribution of,

$$F_t(d) = 1 - \frac{m_r m}{m_n}^{1+\frac{m_r}{m_n}} d + \frac{m_r m}{m_n}^{\frac{2}{1-\alpha}}$$

Moreover, mean field approximation leads to an analytical limiting approximate for the clustering in this network, as measured by the fraction of transitive triples among triples,

$$Cl^{TT} \to \begin{cases} \frac{1}{m(\frac{m_r}{m_n}+1)} & \text{if } \frac{m_r}{m_n} \geq 1 \\ \frac{(m-1)\frac{m_r}{m_n}}{m(m-1)(1+\frac{m_r}{m_n})\frac{m_r}{m_n}-m(1-\frac{m_r}{m_n})} & \text{if } \frac{m_r}{m_n} \leq 1 \end{cases}$$

Dynamic Strategic Networks

Once again, looking at holistic dynamic models, that ignore underlying mechanisms, Lucas' critique kicks in and we might worry about the out of sample performance of random models parametrised for data with different microscopic conditions. Again, the obvious response is to build models in which the macro-properties of the whole network emerge endogenously from the imposed micro-properties of its components. This leads us to consider the strategic choices of individuals in a dynamically forming network environment.

One solution to random models' silence on the small worlds feature of data comes in the form of the Islands-Connections Model of Jackson and Rogers (2005b). This model assumes agents receive utility from being linked to others, that decays exponentially with the length of the shortest path between them, $\ell(i,j)$. That is, an agent receives utility δ from an immediate neighbour, δ^2 from a neighbour of a neighbour, and $\delta^{\ell(i,j)}$ from some agent with a minimum path of $\ell(i,j)$ between them. For the sake of tractability a maximum path length that can provide utility is set as D. With this distance based utility in place, links are given a two-tier cost structure. This is explained intuitively as it costing c to link with an agent on the same *island*, I_i, and C to link with agents on other *islands*, I_{-i}, with $0 < c < C$. This gives each agent a payoff of,

$$u_i(g) = \sum_{j \neq i : \ell(i,j) \leq D} \delta^{\ell(i,j)} - \sum_{j:ij \in g} [c\mathbf{1}(j \in I_i) + C\mathbf{1}(j \in I_{-i})]$$

It can then be shown that pairwise stable networks must involve every agent on each island being directly connected, that diameter is no greater than $D+1$, and that clustering is tightly bounded. However, this relies on a cost structure such that it pays to be

connected directly to agents on the same island, and directly to one agent on another island who is similarly connected. Hence, the model gives a potential explanation for small worlds with high clustering, but both predictions are somewhat obvious from the model setup and represent pretty weak out of sample confirmation of the modelling assumptions, leaving us to wonder how many other of the infinite possible setups might equally reproduce the stylised facts - it would be far better here to match a more specific observed micro-structure with its corresponding observed macro properties.

A starker dynamic analog of the static networks in the previous section is achieved by introducing or deleting edges one by one, according to whether a randomly selected edge satisfies pairwise stability — that is, if it is to be added it offers both nodes improved utility, while if it is already present its deletion offers at least one node improved utility. This process has the downside of becoming trapped in pairwise stable networks from which another pairwise stable network cannot be reached by a single addition or deletion of an edge. To overcome this problem, and ensure that all pairwise stable configurations are possible, Jackson and Watts introduce a trembling hand error, whereby agents' desired action is carried out with some probability, $1 - \epsilon$ and the opposite is carried out the rest of the time ϵ. This randomisation means that pairwise stable networks can be left even when it is not optimal, and allows the whole set of equilibria to be explored (Jackson, 2003).

With every equilibrium reachable, the question becomes the same as for static networks: what utility functions and payoff structures reproduce the stylised facts? However, the dynamics of these properties could not be fully determined by ϵ, which seems likely to produce punctuated equilibrium dynamics like those of similar evolutionary game theory equilibrium concepts — specifically stochastic stability (Young, 2004). The overall validity of the model therefore depends on both deeper knowledge of dynamic stylised facts for networks, and on a fuller investigation of the model's general dynamic behaviour.

Dynamic network models applied to peer-pressure enforcement

A prototypical network formation paper (certainly for my purposes) is that of Vega-Redondo (2006), modelling the emergence of social networks under the peer-pressure enforcement mentioned in chapter 1. It examines first static network equilibria, and then a dynamic formation process, under shifting incentives, by both simulation and mean field approximation. The model is one of perfectly rational agents, for whom a common edge means competing in an infinitely repeated prisoner's dilemma — representing some collective action problem — but also a source of information on the conformity of every agent in the component — though news of defection from the common norm strategy is transferred by only one edge each period. This information transfer service increases conformity of partners to the collaborative strategy, and therefore serves as *collateral* against defection. In its dynamic form, the network grows in a way that borrows from both strategic models described above: at each time period each agent is randomly given the opportunity to form a link with another node within their component of the network or to form a link with any node in the network. Whether or not a link is formed is then determined by whether it offers a Pareto improvement to both agents. With no variation in the idiosyncratic payoffs to the prisoners' dilemma games, the network almost surely converges to the complete network. However, when the payoffs for each link are redrawn each period, with some incremental probability ϵ, any equilibrium network arrived at eventually ceases to be an equilibrium and so the system evolves away from it. The process does not arrive at a fixed configuration, but is ergodic and has an unconditional long run distribution (invariant measure) that can be found by simulation.

In this framework it is found that increased volatility (ϵ) leads to: lower network density, through more depreciated *social capital* (collateral) each period; lower diameter of components, making what collateral there is more robust to collapsing relationships; a smaller largest component, again representing the instability of less dense social capital; lower average payoff, because with less certain value to relationships it pays to

defect more often and so the worst case equilibrium of the prisoners' dilemma is realised more easily.

In contrast to models like that above, where the network's value is partly derived from the information it brings on others' defection, Jackson et al. (2012) proposes a model with full information, where peer-pressure is the only value of common neighbours. The benefit of this simplification is that it allows an analytical solution.

The network formation process has two stages each time period: first, in the manner of the simultaneous move game described in the previous section, agents choose which links they would like to *retain* from the previous period; second, a favour is asked at random, by node i of node j, and if it is not performed then the link ij is removed. As with many repeated game scenarios, one equilibrium is the *grim trigger* strategy wherein all agents maintain links and perform favours until one defects, and then all agents sever all links. This is just one of infinitely many possible sub-game perfect strategies, and so the authors introduce an equilibrium refinement concept of *renegotiation–proof–ness*. This makes strategies robust to reconsideration of the grim trigger after defection, by requiring in any subgame that continuing the *renegotiation proof* strategy (and the network it produces) not be Pareto dominated by some other *renegotiation proof* strategy (and the resulting network).

The authors characterise the actual set of renegotiation proof networks recursively, for a particular cost structure. They then further refine their equilibrium concept by looking for *robust* networks, in which the network arrived at after the grim trigger is only changed locally — that is, networks in which there are no cascades of broken relationships after a single defection. They find that this promotes a unique network structure of complete subnetworks linked in a tree-like structure[8]. These *social quilts* have the high clustering we look for in imitations of the real world phenomena, but the authors find the support measure of density more apt.

As stressed at various points, there are serious drawbacks to powerfully unrealistic

[8]With no cycles as large as the complete subnetworks.

assumptions like perfect information. Nevertheless, the concept of *robust renegotiation proof* networks is an intuitive and powerful extension of the existing game theoretical tools for networks. It should be noted, however, that the tractability of the model is dependent on a growth process that involves only deletion of links. The model is therefore quiet on issues of how networks initially form.

Dynamic models applied to the Economics of recessions

Models such as Vega-Redondo's abstract from real world features that are clearly relevant. As explained in chapter 2, such *heuristical* ceteris paribus assumptions invalidate any policy implications drawn from the model. For this reason the eventual aim of any model development exercise, should allow for all conceivably relevant variables that are found to have an effect — a macro-socioeconomic model. A strong step in this direction comes in the form of Bottom-up Adaptive Macroeconomic (BAM) models, developed as part of the CATS program for computational experiments on Complex AdapTive Systems (Gatti et al., 2011; Delli Gatti et al., 2010; Gatti et al., 2008; Gaffeo et al., 2008; Gatti et al., 2006).

These models could not stand in starker contrast to the RA paradigm of the New Neoclassical Synthesis: they rely on simulation rather than analytical solution; rational choice is dropped in favour of adaptive-behavioural algorithms based directly on survey evidence of actual context specific human behaviour; and most important, they are populated by heterogeneous agents who interact locally. There are obvious drawbacks to behaviourism, when a simple algorithm is used to capture something as complex as the human mind. Still, in fixed roles, that could likely not be described any more dynamically by tractable games, it seems as robust a description as we can get to macroscopic change.

BAM model specifics vary, from firms only with an equilibrium credit market (Gatti et al., 2008) to all markets and R & D being agent based (Gatti et al., 2011), and from fixed network structure (Gatti et al., 2006) to macro structure that emerges from

strategic partner selection (Delli Gatti et al., 2010). What are common features of the models however, are a fat tailed distribution of firms' sizes, a regular cycle for economic output punctuated by periodic cascade events (resembling great recessions), and correlations of typical economic variables that closely match the orthodox economic stylised facts (Gaffeo et al., 2008).

In the latest BAM model (Gatti et al., 2011) network formation follows a general pattern of search for best price among a (mostly) random set of partners: in the labour market, firms post wage offers for fixed term contracts and unemployed workers apply first to their former employer (as long as they were not fired) and then sequentially to a random set by descending wage; meanwhile, in the goods market firms post prices and agents check the prices of an (almost) random subset of firms and buy from that with the lowest price and outstanding stock — they show some preferential attachment by always including the largest firm from their last search in the new set, and this is important to the dynamics. Such schemes might seem *ad hoc*, but when based on actual behaviour, observed over many macroeconomic states, they are robust to the question mark that hangs over the rational choice model's domain of relevance — if not to radically different macroeconomic circumstances. In this way the stylised facts of aggregate fluctuations and co-movements of variables, that are generally pursued by DSGE Macroeconomics, are reproduced reasonably well without any stochastic buffeting of the whole system. Unfortunately, this model still relies on descriptions of markets with a single equilibrium price. As I will explain in the next chapter, even alone this abstraction causes serious problems. A further concern for this research program is the feasibility of simulating populations that approach those of the actual economy: the law of large numbers might be expected to significantly reduce any volatility seen in hundred agent models as the simulation becomes populated by millions of agents.

Hierarchical Networks

Although not exhaustive, this literature review gives a diverse cross section of network and complex models. Several of the dynamic models tried to address the issue of enforcement, from the peer-pressure perspective described in chapter 1. But, none attempted to model the hierarchical enforcement that I have chosen to study in this chapter. This gap in the literature adds value to my modeling effort. But, it does not undermine this literature review, because several useful tools have been introduced in the study of peer-pressure enforcement and simulated economies: powerful descriptions of relationships through graph theory, and simulation techniques to compensate for intractability, to name just two.

3.4 A Hierarchical Enforcement Model

To compete with the inertial resistance of well established theory, any alternative paradigm has to be as accessible as it is effective. I therefore open the innovative part of this paper with an intuitive account of this framework, before giving a precise mathematical formulation.

3.4.1 Intuition for the model

As many of the specific examples in section 3.2 showed, third party enforcement can be (and is) applied in a finer grained way than Hobbes or Locke discussed. Because of the limits to individual enforcers' abilities to monitor and interact with individuals, we tend to see hierarchies of enforcers in almost all institutional settings; so that the enforcement provided by the sovereign is actually the result of their enforcement between sub-sovereigns, who in turn enforce between sub-sub-sovereigns etc.. My aim is therefore to capture the essence of an enforcement hierarchy, in which the failure of some one enforcer can cascade down to significantly affect the aggregate - in the way that a

single bank's collapse might affect the solvency of its client firms, or economic failures in the West might lead to the collapse of its client governments in the developing world. I therefore abstract away from many features of real societies, which, it is hoped, have negligible impact on the overall behaviour. This is made possible by the ubiquity of third party enforcement relationships: anywhere that some decision maker acts as an arbiter of some defectable relationship between other parties, one can describe this as a third party enforcement relationship. Hierarchies arise where the enforcers are themselves subject to the enforcement of others, like the banks just described which are enforcers between lenders and borrowers but are also intermediated by a legal system resting on the government's authority. The firm, a single entity under the old economics, here becomes the hierarchy of cooperative relationships described in the previous section, with each regulated by some manager in the tier above, until a final enforcer is reached, whose relationship with shareholders is enforced by government.

Why should we include all these sub-sovereigns in our description, and not just stick to the sovereign of Hobbes? Omitting the fine granularity of hierarchical enforcement would mean only the sudden losses of all contracts, when the highest sovereign failed, rather than a scalable disturbance. This suggests that abstracting to a single enforcer would not be a negligible assumption, because it would change the behaviour of the model.

The intuition for this scalable volatility arising from the mechanism is simple: as in section 3.2, the third party enforcers only have an incentive to carry out their role if they will lose it otherwise. But, when error or a lack of resources causes them to fail in their responsibility they will still presumably lose their place as enforcer. This volatility will be reflected in output because, if an enforcer is removed then no contracts can be enforced until a replacement is installed, whether those contracts contribute to economic output or some less measurable social outcome. How could aggregate output be affected by a few middle managers failing? Because the enforcers will themselves need contracts enforced, they must also have an enforcer, stretching above them in

a hierarchy. These occasional breakdowns in enforcement, and the resulting loss of productive contracts, deny subordinate enforcers the resources needed to themselves enforce agreements. This can cause their failure too, and give rise to significant aggregate volatility through large *cascades* or *avalanches* of failed enforcement. That is, in a hierarchical society where enforcers enforce contracts between enforcers (who themselves enforce contracts between enforcers...), it is easy to see how one small error could lead to an avalanche of succeeding errors, all adding up to a significant aggregate effect.

The story this model tells is a departure both from the technology and policy driven recessions of monetarist macroeconomics, and from the amplifying price frictions of new Keynesians. Nor are recessions driven by wild over-valuation of goods, compared to some objective measure that somehow escapes the subjective, conventional, nature of value. Instead, recessions are paralleled by massive coordination failures, where formerly productive relationships break down for lack of enforcement against conflicts of interest. Contagion of these coordination failures would be carried by the dependence of other enforcers' potency on the resources from their own enforced relationships. Further, civil unrest and revolution enter the picture: the Arab Spring, and western rioting, formally become part of the same phenomenon as the contemporaneous global recession, rather than an addendum outside the theory. This constant disequilibrium is not necessarily inefficient either (Bak, 1996): the need for the enforcement relationships to collapse in order to prevent malfeasance, could be essential to the value they bring while intact - although elaborating on these fine mechanics is a matter for further research.

Intuitive support for the approach also comes from formal statistical evidence, old and new. *Pareto laws* are distributions with hyperbolic decay of density into their tails and, therefore, infinite variance[9]. These are observed widely in economics, and elsewhere, with income distributions and firm size distributions often argued to follow power-

[9]Infinite variance arises because, as we move into the tail, the squared difference from mean grows more quickly than the density decays.

laws in their upper tails (Gatti et al., 2008). One established mechanism for generating power-law distributions is Herbert Simon's *preferential attachment*, sometimes dubbed "the rich get richer", which is often seen in the network literature. Preferential attachment connotes a situation wherein the growth of a value is proportional to its size. With firms thought of as sub-graphs within the hierarchy, below some highest direct subject of the government, preferential attachment is clearly a relevant mechanism to their size distribution. Indeed, the effect is realised here simply by having an equal probability that new enforcement relationships are formed with every node in the network, so that larger components have a larger probability of being joined.

More recently, evidence has started to build that neither of the traditionally competing views of macroeconomic time series' dynamics holds: variables like GNP are not stationary, and nor are the series of changes between observations. Instead, they are better described by the intermediate representation of *fractional integration* (Mayoral, 2005). Fractionally integrated series are the discreet analogue of what physicists would call 1/f-noise[10]. There are several known models that produce 1/f-noise, but one of the most prominent is the Self-Organised Criticality framework of Bak (1996). In the canonical SOC model and its phenomenal analogue, the Rice Pile, one sees power law distributed avalanches removed from a slowly regenerating whole; this obviously corresponds closely to the structure currently being sketched, and gives more intuitive support for the suggestion that it might replicate macroeconomic dynamics.

Crucially, for the argument that societal volatility can be described by the same mechanism as Economic fluctuations, cutting edge sociological research also suggests fractionally integrated time series for conflict data. This, soon to be released, work is based on the GDELT discrete event data set, a coded record of global news from the last 30 years, with over 200 million political events. Some of the events recorded are scaled descriptions of conflict across the world, be it civil strife or war. The time series of these events' severity, for particular countries, shows the same fractional integration

[10]This is a signal which, in the frequency domain, shows a power law decay in its power-spectral density (Mandelbrot and Van Ness, 1968).

behaviour as GNP. It will be an important supplement to this research to see whether the precise properties of that series compare to economic fluctuations.

To recapitulate my conclusions from chapter 2: with the shortcomings of the conventional microfoundations, an attempt should be made at a new reductionist model for macroeconomics; this should start with a description of behaviour, for components of the economy, that has already proved an efficient way to capture information, and predict continuity. What I am proposing here is to take for that purpose the well established and useful descriptions of third party enforcement and hierarchical social structure. These are descriptions so useful that they transcend scientific, or even technical, discourse and are used in everyday language.

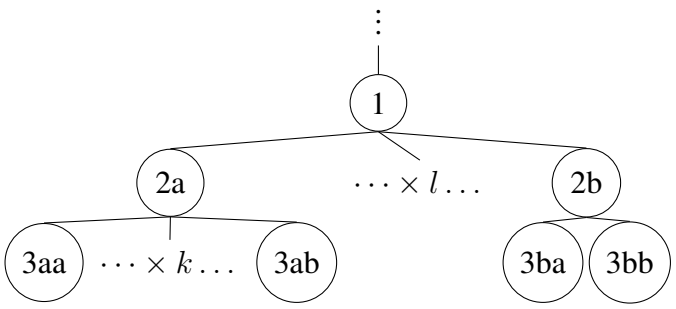
3.4.2 The formal model

The model is populated by a set of agents N, in a *directed* network g — so that a link from agent i to agent j, ij, has a different meaning from a link from agent j to agent i, ji. Each agent, $i \in N$, can act as an *enforcer* to any other agents $j \in N$, $ij \in g$, and as an *enforcee* to any one agent $k \in N$, $ki \in g$, meaning that k enforces i's productive contracts with other agents. The number of direct enforcees that each enforcer is capable of sustaining is limited to some positive integer, m.

Enforcement is transitive: if i is an enforcer to j and k, $ij, ik \in g$, while l and m are enforcees of j and k respectively, $jl, km \in g$, then l and m's productive relationship is enforced. This means that a productive relationship will exist between every pair of enforcees in a *component* — a particular hierarchy $g' \subset g$ wherein every two agents share an enforcer either directly or indirectly, and share none with any agent outside of g'.

Figure 3.2 shows a hierarchical enforcement structure graphically. Let us compare this to two of the examples from section 3.2, a firm and a drug cartel (as a black market institution). In our firm node 1 represents the general manager of Clips stationary.

Figure 3.2: A hierarchical enforcement network



Node $2a$ is the head of accounts, and $2b$ is the head of IT, while all the other nodes connected to 1 are other department heads. The $3xx$ nodes are subordinates in these two departments. In the case of the drug *cartel*, 1 represents some lieutenant in the drug cartel, with a drug lord somewhere above them (along the dotted line). $2a$ represents a buyer, with $3aa$, $3ab$, and others as producers that they coordinate. $2b$ and the other $2x$ nodes are local distributors, with $3ba$ and $3bb$ as the street level dealers that they coordinate.

Time is discreet, and every period a *network growth event* takes place, consisting of a series of steps:

NG.1 Enforcee search — an agent, $i \in N$, is chosen to be offered a new enforcer, uniformly at random from those agents with no relationships;

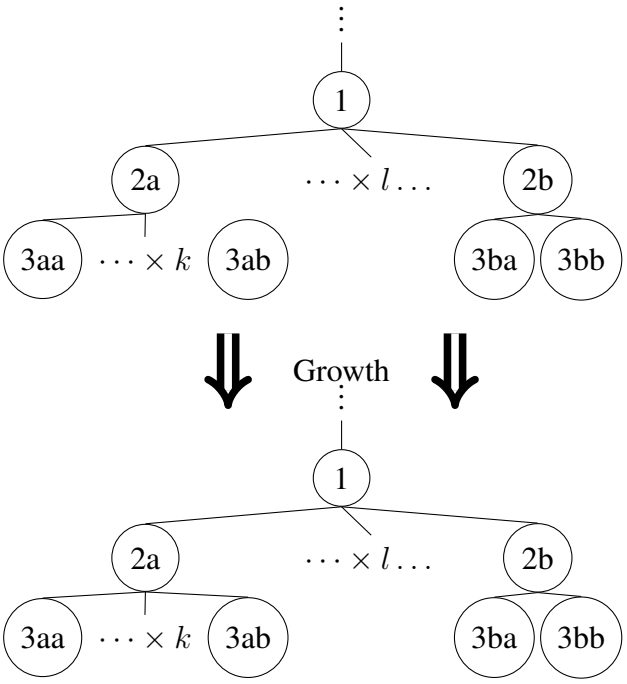
NG.2 Enforcer search — another agent, $j \in N$, is chosen uniformly at random to be offered as an enforcer; while j has m or more enforcees, j is redrawn from N uniformly at random;

NG.3 Relationship formation — ji is added to g.

The value of every productive relationship is assumed to be of the same order. Hence, within any one hierarchical component, C, the production taking place, y_C, has the same proportional relationship to the number of pairs of enforcees in that hierarchy:

$$y_C \propto n_C \times (n_C - 1) \text{ where } n_C = |\{j \in C | \exists ij \in g\}|$$

Figure 3.3: A network growth event, between top and bottom



Assuming that the largest component is the formal economy, this then makes aggregate production, y, simply the production of that component:

$$y = \sum_{C} y_C$$

Figure 3.3 illustrates this growth process: as time progresses between the left and right hand figures the formerly disconnected node, $3bb$, is offered $2b$ as an enforcer. Since $2b$ has fewer than m enforcees, this relationship is allowed.

In the context of our first example, $3bb$ is a potential new employee for the IT department. After coming to trust the IT manager's regulation of their interaction with other employees, $2b$ contributes their full effort in collaborating with the other members of the firm, and adds to its productivity by the number of other employees whose productivity they are enhancing. Meanwhile, in the context of our second example, $3bb$ is a budding street-level dealer who starts paying for the protection of $2b$. As part of the same component, under the enforcement of the single drug lord, there is now the

infrastructure in place for producers' output to be supplied to $3bb$ without fear of non-payment or supply interruption. Moreover, now that there is an institution by which to settle territorial conflicts with other dealers, the productivity of all the other $3x$ dealers is increased. In this latter case, the nodes would all be part of some component other than the largest, which is the formal economy.

At a fixed interval, τ, a *network collapse event* takes place, consisting of a conditionally repeated series of steps:

NC.1 Enforcer failure — an agent, $i \in N$, is chosen to fail, uniformly at random;

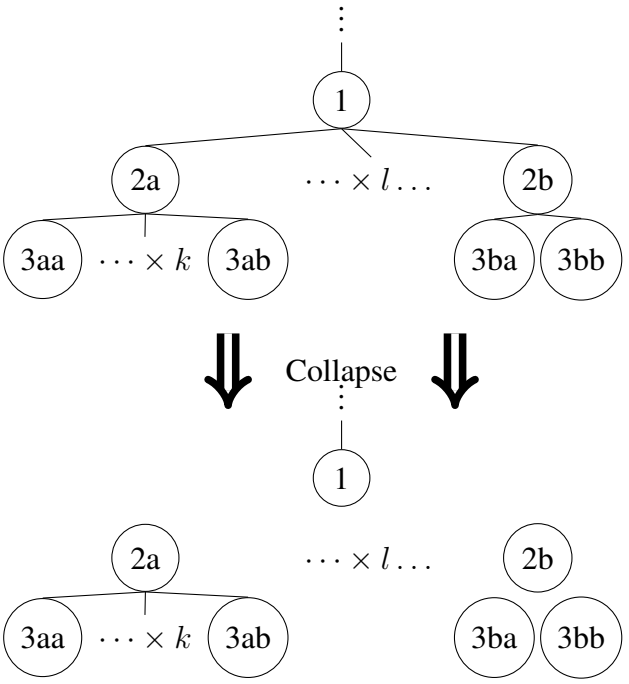
NC.2 Enforcee severance — with a certain *contagion probability*, p, each link between i and its enforcees, $N^i = \{j \in N \mid ij \in g\}$, are removed from g;

NC.3 Contagion — if any links were removed in the previous step, $|N^i| > 0$, all steps are repeated for every other agent adjoining those links, $j \in N^i$, in order to capture the immediate contingency of their ability to enforce on the value from their own enforced relationships.

Figure 3.4 describes a Network Collapse event. As 1 fails, it becomes unable to enforce between the other nodes below it. A fraction of its $2x$ enforcees, that on average would be p, then also fail. In this instance $2a$ is one of those failures. $2a$ is then unable to enforce between pairs of nodes beneath it, and these relationships' productivity is lost to the economy.

In our first example, of a firm, we can think of the general manager having some random disturbance in their private life. Let's say that they start to go through a divorce, so that they spend less time on their job and are generally distracted. If this prevents them from effectively enforcing the interaction of the different departments below them, then these departments will no longer trust their contribution to the firm's output to be properly recognised, and they cease to cooperate. Now that the accounts manager, $2b$, cannot get IT to support the accounts staff, those staff lose trust in that manager's

Figure 3.4: A network collapse event, between top and bottom



efficacy. Without trust in their manager's enforcement, the accounts staff worry that their work won't be properly attributed, and that their colleagues will not put in full effort; they stop contributing, and the firm's output drops.

In drug cartel of the second example the failure of the lieutenant, 1, could represent their assassination by a member of a rival group. With the loss of the lieutenant's protection, the local distributors have no means to settle their disputes with one another. $2b$ is killed by one such, and can no longer enforce relationships between the street level dealers beneath them. With conflict breaking out between these street level dealers, the productivity of this part of the black economy falls proportionally to the number of conflicts breaking out.

In keeping with my methodological prescription from the previous chapter, this model is based on very successfully used descriptions of society. It adds very little else to this, and so comparison to other mathematical models is not particularly necessary. The greatest influence from a mathematical model, is that of the Self Organised Criticality

paradigm described in the previous subsection, due to Per Bak (Bak et al., 1993; Bak, 1996). It should be noted, however, that other than allowing for the obvious collapses in third party enforcement I take nothing from this literature. Specifically, my abstractions have not been tailored to mimic the mechanisms found to underlie SOC by Vespignani and Zapperi (1998).

3.4.3 Robustness Check: Swapping Enforcers

In chapter 2 of this thesis I concluded that a reductionist model of the economy was only valid if the descriptions of its components could be made more realistic without changing the model's qualitative behaviour. Because a model with complete detail of all components of the economy is not feasible, this criterion can never be assessed outright. What can be done, as I did with DSGE models in the previous chapter, is to show when a modelling assumption fails this criterion. To begin applying the same rigours in my own research programme, I also test a version of the model in which one of the obvious abstractions is relaxed, namely the fact that enforcees never swap their enforcer. In observing hierarchies in the real world, it is plainly the case that individuals swap from being enforced by one third party to another. For example, people frequently move between and within firms, swapping one manager for another. Between network components too, we see individuals swapping enforcer, like those moving between the formal and informal economies. The formal model described in this section abstracts from this by assuming that an individual will only take on a new enforcer after a collapse event, in the hope that the effect is trivial. To check whether it is indeed trivial, I introduce a variant of my model in which agents who already have enforcers are offered a new enforcer, and choose whether to swap. A difficulty here is describing the way in which agents choose between enforcers. A very rich description of the benefits of having different enforcers would reduce both the model's generality and its parsimony. Instead, I keep things simple by making an observation about real world hierarchies: the higher up a given enforcement hierarchy,

is an agent, the less likely they are to leave in favour of another. We see this both in rogue states and criminal organisations, where the high-ups are committed to their position while the lowest members of the hierarchy often chose to move over to a the larger hierarchy of the international community or formal economy. Indeed, that this observation can be made about two different scales of hierarchy helps us to specify the conditions for swapping. The variables determining the swap here are the size of the enforcement hierarchy, and the position in the hierarchy. In order for the relative value of these two to determine agents' choice of hierarchy regardless of the size of the hierarchies in question, the importance of each must grow at the same rate. Because the number of agents in these tree-like enforcement networks will grow exponentially in the number of levels, with base m, the importance of hierarchical position must be of the order of $m^{\text{hierarchical position}}$. So, were agent $3aa$ in 3.2 to weigh up agent $2a$ as an enforcer against agent 1, they would compare the ratios $\frac{\text{component size}}{m^2}$ and $\frac{\text{component size}}{m}$ respectively. Obviously, they would choose to have agent 1 as enforcer here because they get closer to the top of the hierarchy, while still having the same number of other agents with whom they can make enforced agreements, because they remain in the same component. Intuitively this importance of hierarchical position makes sense, because third party enforcement is dependent on asymmetrical power between enforcer and enforcee, and we can imagine easily enough that agents prefer more power.

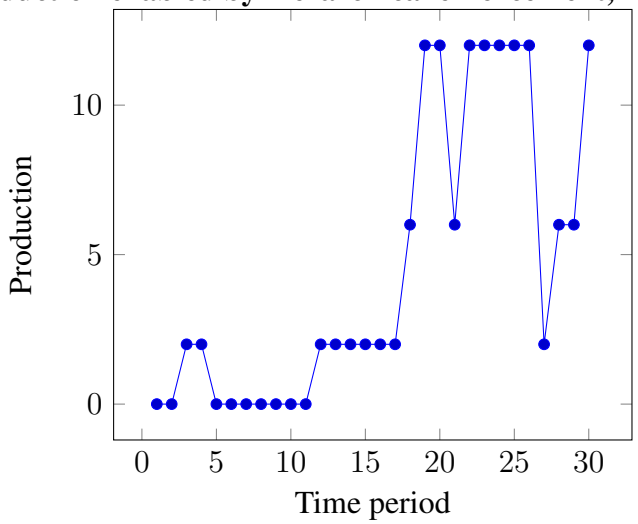
Put simply, the condition for swapping is described in step 3 of a revised *Growth Event with Swapping*:

NGS.1 Enforcee search — an agent, $i \in N$, is chosen to be offered a new enforcer, uniformly at random from those agents with no relationships;

NGS.2 Enforcer search — another agent, $j \in N$, is chosen uniformly at random to be offered as an enforcer; while j has m or more enforcees, j is redrawn from N uniformly at random;

NGS.3 Enforcer choice — if i already has an enforcer, $k \in N$, i swaps j for k, and proceeds to the next step, only if this would increase the ratio $\frac{\text{component size}}{m^{\text{hierarchical position}}}$;

Figure 3.5: Production enabled by hierarchical enforcement, initial behaviour



NGS.4 Relationship formation — ji is added to g, and ki is removed from g.

The empirical testing of this second modelling set up, as a robustness check on tests of the first model, are described in section 3.6.3.

3.4.4 Sense Check: Small population model behaviour

Were this model to run with a realistic number of agents, it would be nigh on impossible for a human to keep track of the path of any individual agent. This is the essence of Complexity Science: regular aggregate patterns, despite our being unable to follow regularity in the individual components. But, in order to demonstrate some model behaviour we can run it with a very small population of agents, for a relatively short number of periods. Table 3.1 and figure 3.5 show exactly this. The model has just 10 agents, a maximum of 3 enforcees per enforcer, an enforcer failing every 2 periods, and a 95 percent chance of failure causing failure of each sub-enforcer.

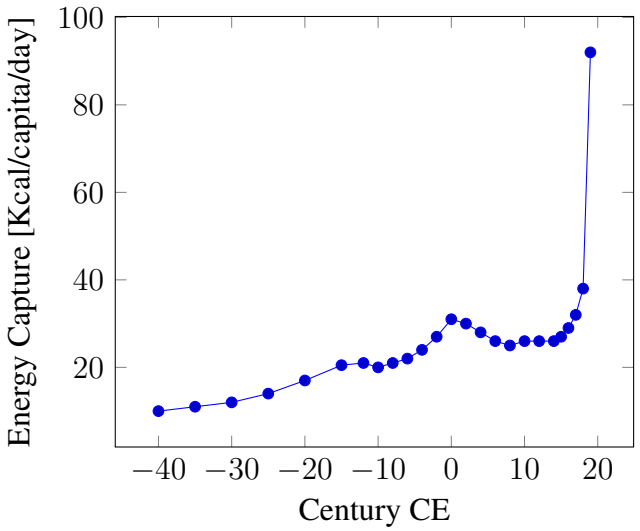
Although the coarse granularity at this scale makes for a very abrupt step process, we already see the behaviour we might hope for in the properly callibrated model. There are periods of steady growth, as agents find enforcers and hierarchies develop. Each

Time	Production	Commentary
1	0	No pairs of enforcees share an enforcer.
2	0	...as before...
3	2	Two enforcees now share an enforcer.
4	2	...as before...
5	0	Their enforcer fails.
6	0	No pairs of enforcees share an enforcer.
7	0	...as before...
8	0	...as before...
9	0	...as before...
10	0	...as before...
11	0	...as before...
12	2	Two enforcees now share an enforcer.
13	2	...as before...
14	2	Any new enforcement is being...
15	2	...countered by collapse.
16	2	...as before...
17	2	...as before...
18	6	Three enforcees now share an enforcer.
19	12	Fourth enforcee joins largest component.
20	12	Only smaller components grow/collapse.
21	6	A sub-enforcer has failed!
22	12	The freed enforcee was replaced.
23	12	Only smaller components grow/collapse.
24	12	...as before...
25	12	...as before...
26	12	...as before...
27	2	Enforcer of main component fails!
28	6	Three enforcees now share an enforcer.
29	6	Only smaller components emerge/collapse.
30	12	Fourth enforcee joins largest component.
Best freq.	1	

Table 3.1: Initial production levels of model with population of 10 agents

additional agent joining the largest component (the formal economy) means an extra unit of trade for every existing enforcee. But, periodically one of the enforcers in a hierarchy fails, as do many of the sub-enforcers below them in the hierarchy. This leads to sudden drops in production, like recessions striking a real economy.

Figure 3.6: Energy Capture over 60 centuries (relative to Contemporary Era), as a measure of socio-economic activity.



3.4.5 Reality Check: Complete Social Collapse

One way in which we could immediately dismiss the model, is to consider whether any of the more extreme behaviours it describes are completely out of keeping with what we have previously observed about the Macroeconomy. The most extreme feature of the model, is its ergodicity: once a hierarchy has grown to include every single agent, which can happen with finite probability, there is then a finite probability that the highest enforcer is subject to a collapse event so that all enforcement relationships are broken. Such a collapse of all social order is an extreme prediction and we have to compare it to what we have observed, to see if there is any precedent for such a property. As it happens there are (at least) two such massive social collapses in world history, neither of which yet has a conclusive explanation: the fall of the Western Roman Empire, and the Bronze Age Collapse.

As an indicator of general social development Morris and Farrar (2010) estimate the gross amount of energy being captured by Western Civilisation at various points since 14000BCE. The evidence ranges from modern statistical digests to literary accounts of farming, industry, and lifestyles, to archaeological evidence for diet, crafts, and quality

of life, to ice core evidence of Roman era industrial polution. Figure 3.6 shows this measure of the progression of the western world, and two obvious and massive drops can be seen. The first, around 1200BCE, is known as the Bronze Age Collapse. It corresponds to an arguably sudden transition in the western Mediterranean, from a set of large highly interrelated states connected by robust trade routes to isolated villages using only local resources — a period known as the Greek Dark Ages. The second, more significant, drop represents the collapse of the Western Roman Empire, wherein a massive military and trade infrastructure that spanned much of Western Europe dissolved over a relatively short period of time, leading to the Dark Ages in Europe.

This digression into history is, of course, not active support for the model. It is simply the failure of an obvious criticism. In the next section I develop a formal method for assessing whether mine is a reasonable description of macroeconomic data.

3.5 Empirical Method

Tractability of aggregate non-equilibrium dynamics is rare, more so when they are expected to display long-range dependence –because the degrees of freedom bearing on the state of any constituent element become too numerous to track individual components. For this reason, any empirical assessment methodology, used to test the model, will have to be compatible with simulated data. The recent growth in computational approaches to DSGE models has brought with it empirical methods designed to cope with exactly this problem, and here I adopt a prominent example, Indirect Inference.

Although I find that my model does indeed generate long memory in its output, I cannot show this analytically. Not understanding *Why* long memory is generated is, of course, limiting. But, as explained in section 3.3.2, it is common in Network and Complexity Science to accept this limitation. After all, it is often worth sacrificing some of our understanding of a model, in order to better understand the relationship of the model to empirical evidence.

3.5.1 Indirect Inference Testing

The inference of models' conformance to an observed phenomenon is said to be *indirect* when it is judged through the similarity in descriptions of both according to some other, *auxiliary*, model. For example, in Le et al. (2011) the model being tested is a DSGE variant and the auxiliary model is a Vector Autoregressive Process on a standard set of macroeconomic variables. The stochasticity of the tested model means that a distribution of different time series will be generated by the model for different realisations of the random variables. This distribution will give rise to a distribution of the parameters chosen to best describe these time series in VAR terms. Meanwhile, a VAR fitted to the actual observed macroeconomic data will have specific parameter values. The higher the likelihood of seeing these observed parameter values under the distribution generated by the model, the more credence is lent to the model.

To summarise Indirect Inference testing in step-by-step form (Le et al., 2010):

IIT.1 — Estimate the auxiliary model for the observed data;

IIT.2 — Repeatedly simulate the economic model, and estimate the auxiliary model on the simulated samples to produce a distribution of the auxiliary parameters;

IIT.3 — Compute the *Wald Statistic* –the squared difference between observed and average simulated auxiliary model parameters, scaled by the simulated parameters' variance –and compare this to a χ^2 distribution with number of degrees of freedom corresponding to the number of parameters.

It would be unusual to think of macroeconomic data itself as the object of interest in economics. Therefore, this approach is not unlike the regular comparison of a theoretical model to an unobserved data generating process, by comparison of the data produced by each, in that some information is acknowledged as already having been lost. Indeed, comparing an arbitrary number of moments as the auxiliary model would

be equivalent to a more traditional likelihood test. If our objective is prediction of future phenomena[11], then this loss of information needn't be cause for concern, so long as we believe the retained information is all that which is likely to recur in those phenomena. Unfortunately, VAR-based descriptions have been shown repeatedly to fail at capturing any persistent features of macroeconomic data (Wieland and Wolters, 2011) – that is, to forecast behaviour outside the observed sample. The evidence for fractionally integrated processes better describing economic fluctuations (Mayoral, 2005) makes them preferable as auxiliary models.

3.5.2 Indirect Estimation through Simulated Annealing

The microscopic observations that lead to the structure of my model were uninformative as to its parameters' values. In order to test only the model's general ability to reproduce observed behaviour, it is reasonable to give it the best possible chance, by choosing parameters that bring it closest to that behaviour. Staying close to our Indirect Inference method of appraisal, we can compare an auxiliary representation of the data to the auxiliary representation for simulations of different parameterisations of the model, to judge which is best.

To summarise Indirect Inference estimation in step-by-step form:

IIE.1 — Estimate the auxiliary model for the observed data;

IIE.2 — Simulate the economic model for various parameterisations, and estimate the auxiliary model on the simulated samples to produce a range of auxiliary parameters;

IIE.3 — Select the parameterisation that minimises the distance between the observed auxiliary parameters and the simulated auxiliary parameters.

[11] As opposed to, say, the self-satisfaction of fitting the observed phenomenon neatly into our existing set of models.

In practice, one must choose the range of parameter values for which one simulates, and one must necessarily do so without prior knowledge of whether the best of this range will be close to the best possible. The greater the range, of course, the more likely that the best is close to the Best, BUT the more costly will be the process. The practical solution is to have an algorithm try successive parameterisations, basing its exploration of the parameter space at each stage on what has been found so far. For a distance between theoretical and observed auxiliary parameters that is a continuous function, one cannot get to a globally smaller distance without passing through locally smaller distances. This recommends a *greedy* algorithm, which explores close parameterisations and only progresses to one closer to the observed auxiliary parameters for the next step.

Of course, local minima may exist, separated from the desired global minimum by parameterisations that yield a larger distance. In order to avoid getting stuck in such a shallow valley, one can modify the greedy algorithm to occasionally choose similar parameterisations that in fact increase the distance between simulated and observed auxiliary parameters. This would be what is called an *epsilon greedy* algorithm — the "epsilon" representing a small probability of error. Of course, for this algorithm to ever settle on a particular minimum, the epsilon would have to fall to zero at some point. It is in this modification that we see the spirit of an algorithm class called *Simulated Annealing*.

Simulated Annealing has its roots in the statistical mechanics of metallurgy, as the name suggests. Kirkpatrick et al. (1983) were the first to draw parallels between the behaviour of equilibrium systems with many degrees of freedom, and multivariate optimisation. In metal annealing the temperature is lowered only slowly, so that every part of the system approaches order at the same time and the configuration is globally ordered (minimum possible energy). The alternative is rapidly cooling and becoming locally ordered in small subsets but disordered between these subsets (a local minimum in the space of all system states). This rapid cooling is analogous to the simple greedy

algorithm above, but for an algorithm to match the slow cooling it would need first to allow substantial movement across the optimisation space (like the easy transmission of atoms and local temperature through a hot substance), and slowly reduce this movement to allow stabilisation at the optimum (like the substance settling into large crystals). The epsilon of the epsilon greedy algorithm is therefore reduced as if it were the tendency of atoms to change state under falling temperature.

To summarise Simulated Annealing in step-by-step form:

SA.1 — Randomly choose a neighbour of the current state;

SA.2 — Evaluate the objective function at that neighbouring state;

SA.3 — If the neighbour improves the objective function, then it becomes the current state; if it does not improve, then it may still become the current state with some probability dependent on the system *temperature*;

SA.4 — The temperature is reduced according to the number of iterations thus far.

3.5.3 Fractional integration and Whittle estimation

Having recognised that unfiltered economic data is likely better described by a fractionally integrated process, I here give a brief overview of that class of models. This is followed by a description of the popular Whittle estimation technique for fractionally integrated time series, and other AutoRegressive Fractionally Integrated Moving Average models.

Fractional integration is a generalisation of the more conventional idea of an integer-order integrated time series,

$$y_{t+1} = y_t + \epsilon_t \;\Leftrightarrow\; (1-L)y_{t+1} = \epsilon_t \sim N(0, \sigma^2)$$

It relies on fractional summation, wherein the contribution of summands is reduced according to their distance from the latest in the series:

$$(1-L)^d y_{t+1} = \epsilon_t \sim N(0,\sigma^2) \mid d \in (0,1)$$

In this way, the effects of past random innovations, $\{\epsilon_s \mid s < t\}$, are not permanent but rather highly persistent. Indeed, while $d \in (0, 0.5)$ the correlation function for the process is fixed, but decays hyperbolically:

$$\rho_k \propto k^{2d-1}$$

US GNP has previously been proposed to have a difference parameter in the range $(0.5, 1.0)$ for which second moments are not defined, but reversion to the mean can be expected.

The Local Whittle semi-parametric estimation method is a frequency domain based estimator for d, that is both strongly consistent and asymptotically efficient (Dahlhaus, 1989). It is based on the computationally efficient Whittle Approximation of Gaussian likelihood[12], which makes use of the similarity between a time series covariance matrix's eigenvectors and the frequencies of the Fourier series representation of a time series. But the orthogonality of these frequencies allows the procedure to focus only on the low frequencies, determined by the difference parameter, and remain agnostic about short run dynamics — thereby giving a semiparametric estimator (Shimotsu, 2010). In its most basic form, it relies on the minimisation of the discretised function,

$$Q_n(\zeta) = \frac{1}{2n}\sum_{j=1}^{n-1}\left(ln f_X(\omega_j,\zeta) + \frac{I(\omega)}{f_X(\omega,\zeta)}d\omega\right)$$

with respect to the parameters ζ, for the spectral density function f_X on frequency component ω, with periodogram $I(\omega)$. In order to compensate for non-stationarity

[12]This is, of course, appropriate for non-Gaussian stationary stochastic processes because of their Wold Decomposition.

of the data, along with the unknown mean and a potentially polynomial time trend, I implement a modified version of the above developed by Shimotsu (2010). This Two-Stage Exact Whittle Estimator is efficient for a greater range of difference parameters than any other, $d \in (-0.5, 2.2)$, a range which easily encompasses any proposed in the literature for macroeconomic time series. The conventional Whittle Approximation employs an estimate of the periodogram, which is poor for non-stationary time series, but the Exact Whittle Approximation, used in the second stage of the two step estimator, corrects for this with an exact manipulation of the periodogram rather than an approximation. This leads to the minimisation of the following objective function, for frequencies λ and $\Delta^d X_t = (1-L)^d X_t$,

$$\hat{Q}_n(G,d) = \frac{1}{n}\sum_{j=1}^{n}\left(ln(f_{\Delta^d X}(\lambda_j)\lambda_j^{-2d}) + \frac{I_{\Delta^d X}(\lambda_j)}{f_{\Delta^d X}(\lambda_j)}\right)$$

A preceding step is needed, estimating the differencing parameter, because this exact estimator requires a known mean and time trend. This preceding step uses the tapered Whittle estimator of Velasco (1999), which tapers the time series to remove long memory before estimating the fractional difference parameter. Although valid for the same range of parameters as the exact, this estimator is not efficient which makes the second stage necessary. With this estimate in place however, a suitable estimator of the mean can be selected from between the time series' mean (for $d < \frac{3}{4}$) and its first observation (for $d > \frac{1}{2}$).

3.6 Empirical Performance of the Model

3.6.1 Data

The observed data used for US Gross National Product is taken from the extensive collection of GNP time series compiled by Maddison. This time series runs from 1870 to 2008. Maddison was chosen precisely because fractional integration implies

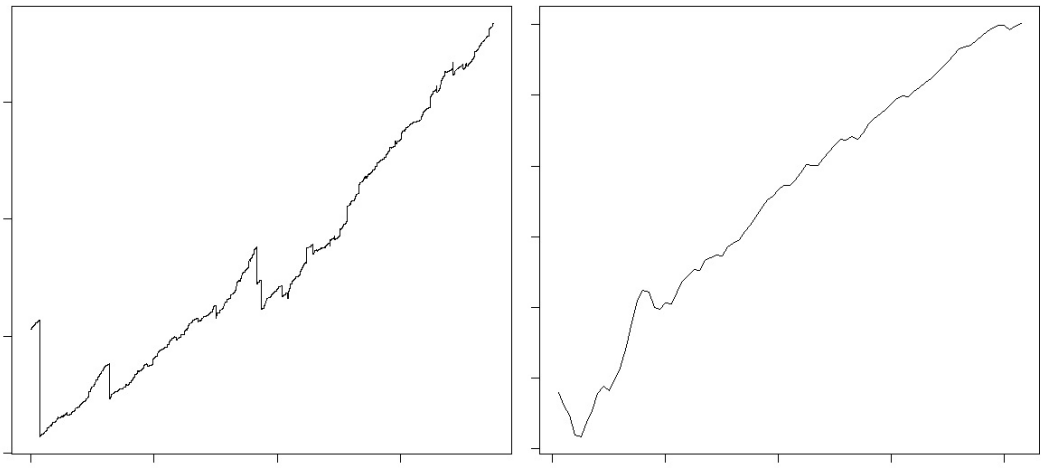


Figure 3.7: Time series behaviour of: simulated data from the hierarchical enforcement model (*left*); and observed US GDP from Maddison (*right*).

interdependence of observations over very large time separations, which makes using data with the largest reliable time separations important for the accurate estimation of the difference parameter.

For visual comparison with the observed US GDP data in the right hand pane of figure 3.7, the left hand pane shows the behaviour of a selected sample from a simulation of the model with an arbitrary parameterisation[13]. One notes the same upward trend and saw-tooth pattern. This sample was chosen specifically to imitate the behaviour of the observed sample, and cannot necessarily be considered representative. Nevertheless, the broad description I am aiming for with this model need not have the observed data as a representative sample either.

The simulated data used for the actual testing comes from ten thousand simulations of the model over twenty thousand periods, with a population of ten thousand agents. These numbers were chosen to make the computation feasible, but are in line with simulation numbers from the DSGE literature, such as in Le et al. (2012). Clearly ten thousand is a much smaller population than the US economy, and one might expect the behaviour to have subtle differences for a larger population. But, the Law of Large

[13]This simulation was over ten million periods with $m = 10$, $\tau = 5$, and $p = 0.95$.

Numbers should be less applicable because of the highly interconnected nature of the population.

Because there is a finite probability of the collapse of the whole network, should the highest enforcer fail, the process is ergodic, and so I considered it unproblematic to start all simulations from an empty network. As the process is intended to exhibit long memory, however, the initial state should be expected to influence behaviour for many periods. To allow this initial state to become trivial, the first ten thousand observations of each sample were burned. The population size was chosen for the burn period in order to give every agent time to acquire an enforcer. The the remaining data was then sampled for estimation of the difference parameter. There is nothing in the microscopic mechanism I have described that specifies a time scale, and so the sampling was done at different frequencies to generate a set of different time-series from each simulation run, with different intervals between observations. Difference parameters were estimated for each of these sampling frequencies, but this is expensive, so sampling frequencies were explored by doubling the interval each time: the first time-series kept every consecutive observation, the second skipped every other, the third took every fourth observation, etc. Sample size was kept constant, and equal to size of the observed sample. This limited the maximum sampling interval to the closest integer to $\log(10000)/\log(2)$. It also meant that samples with shorter intervals did not span the length of the unburned simulation data. In order to maximise the burn period, samples were therefore taken from the end of the initially unburned sample of ten thousand.

3.6.2 Results

Applied to the current problem Simulated Annealing searches one point at a time, through a subset of a grid in the parameter space — which is three dimensional. The size of the grid is chosen somewhat arbitrarily, based intuitively on the duration for each simulation and acceptable exploration time. The parameters controlling the maximum number of enforcees and the interval between collapse events were allowed to

change by one at each step, while the contagion probability was allowed to change by 0.01. Specifically, the algorithm then becomes:

IISA.1 — One of the six neighbouring parameterisations on the grid is chosen at random;

IISA.2 — Ten thousand simulations of the model with these parameters are carried out. The distribution of the difference parameter for this parameterisation is estimated using the Whittle Estimator on the simulated data, sampled at frequencies that are powers of two;

IISA.3 — The difference parameter for the observed data is compared to these distributions; If the highest p-value is greater than that for the previous parameterisation, then this parameterisation becomes the new state for the algorithm; if it is not greater, then it still becomes the current state with probability $e^{-\frac{\text{improvement}}{\text{temperature}}}$;

IISA.4 — The temperature becomes the initial temperature divided by the number of iterations thus far.

I also modify the algorithm to at all times keep track of the global best so far, in case the system jumps out of the global minimum before the complete cooling.

This annealing algorithm selected a parameterisation such that: each enforcer could support at most five enforcees, $m = 5$; Network Collapse events happen at the same frequency as Network Growth events, $\tau = 1$; and the probability of a failing enforcer losing each enforcee was close to unity $p = 0.99$, leading to long cascades.

It was unexpected that higher sampling intervals would display less mean reverting behaviour, as is described by difference parameters closer to unity in table 3.2: major collapses should be more likely over a longer timeframe, while the steady growth of the network should be present at all sampling frequencies. This feature of the tabulated statistics suggests that once the network is fully grown, after the burn in period, the

Sampling Interval	$E(\bar{d})$	$VAR(\bar{d})$
1	0.81	0.12
2	0.90	0.08
4	0.97	0.03
8	0.98	0.01
16	0.99	0.01
32	0.98	0.01
64	0.95	0.02
Best freq.	1	

Table 3.2: Moments of the long memory parameter for different sampling intervals.

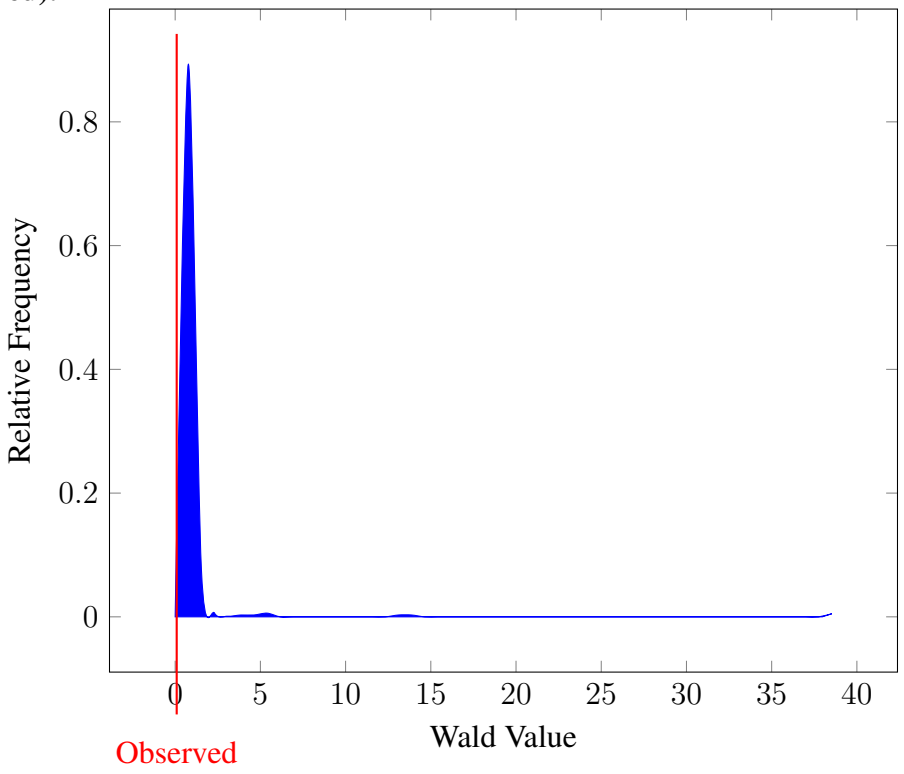
agents who lose enforcement often join a new enforcement network that is large enough to quickly make up the fall in production.

Data series	$\bar{d}$	$var\left(\bar{d}\right)$
Observed	0.846	n/a
Simulated	0.810	0.123
Wald stat.	0.011	
p-value	0.99	

Table 3.3: Moments of the long memory parameter for optimised simulated data, with Wald statistic and p-value for the observed difference parameter.

Table 3.3 shows the results of the Indirect Inference hypothesis test for the optimised parameterisation; the model can clearly not be rejected as an efficient description of the data, with the set of more extreme values having very small measure under the null. As can be seen from figure 3.8, the Wald value for the observed data falls extremely high up the single tail of the asymptotically-Chi-square distribution. It should be stressed that this was not a foregone conclusion, as mathematical models producing long memory are by no means commonplace. For instance, a representative agent equilibrium model such as those employed by mainstream economics should not be able to produce this long memory endogenously. This is because these models do not involve the aggregation of many different units, seen in every other model generating long memory —as described in chapter 4. Long memory could likely only be introduced into a representative agent model by introducing it directly into the stochastic

Figure 3.8: Wald distribution for optimised parameterisation, with observed value (red).



processes describing exogenous factors.

A word of caution is needed, however, because the objective of this chapter was never as ambitious as providing a decisive replacement for DSGE. The results suggest only that the heirarchical enforcement description is compatible with one feature of the macroeconomy. Whether it is the best model for the job of managing aggregate volatility, or the minimal description in the language of chapter 2, is a far more expansive question.

Such a resoundingly positive result from a model that was only a first sketch, should also prompt caution about the methodology used. The large variance of the difference parameter over simulated samples means that many values could perhaps be described by this model, including non-mean-reverting processes with very different long run behaviour. This clearly makes the result less decisive. Moreover, the method of Indirect Inference is innovative, but that means it is still subject to many questions regarding

its real world effectiveness. In particular, the power of the method to reject *false* hypotheses (i.e. bad descriptions) is reasonable for three variables jointly (Le et al., 2012), but is unknown for single variable series. The established results are also only for a stationary DSGE model, and so the power in non-stationary situations is also unknown. It is these questions of power that I will pursue in the next chapter.

3.6.3 Empirical Robustness Check

As discussed in section 3.4.3, I also test a second version of the model which allows the more realistic feature of agents changing enforcers. Simulated annealing again chooses the parameterisation $m = 5$, $\tau = 1$, $p = 0.99$, but this time at a lower sampling interval of 8 periods.

Data series	$\bar{d}$	$var\left(\bar{d}\right)$
Observed	0.846	n/a
Simulated	0.889	0.062
Wald stat.	0.030	
p-value	0.99	

Table 3.4: Moments of the long memory parameter for optimised simulations of the model with swapping.

Once again, as 3.4 shows, the p-value for this parameterisation is very close to unity, and we can absolutely not reject the model as a good description of the data's long memory properties.

This represents the failure of a first attack on the reductionist credentials of this model. But, as explained in section 3.4.3, there can be no decisive confirmation of these credentials and the model should continue to be tested with relaxed abstractions wherever feasible.

3.7 Conclusions and extensions

In this chapter I presented a model that tries to reproduce the macroeconomic fluctuations generally misnamed the "Business Cycle". Rather than being a description based only on observations of a macroeconomy, it is a *reductionist* model that draws on information from the components of the economy to strengthen our evidence base. The information in question is an abstract description of the hierarchical structures underlying third party enforcement of agreements, including the productive agreements underlying the formal economy. My empirical investigation shows that this model can provide a good description of a long sample of US GDP data.

In the previous chapter I made some strong prescriptions for Scientific method. Specifically, I suggested that:

1 new models should be based on successful descriptions from other domains

2 parsimony should be the objective for modelling[14]

3 reductionism was only valid when the description of the components made no abstractions that changed the model's behaviour

The model presented here follows (1) by employing the very widely used social descriptions of third party enforcement and hierarchical structure components of the economy. It attempts to satisfy (2) by describing (long memory) statistical properties of the data, that conventional models do not, while using these component descriptions with all of the other detail of their circumstances stripped away. Obviously, this abstraction is potentially antagonistic to (3); it is an important further question for this new research programme, whether relaxations of that abstraction preserve the model's general behaviour. This is a condition that the conventional Macroeconomic modelling paradigm has been shown to fail in many ways (see Chapter 2), and it remains to be seen whether my attempt is any better in this dimension.

[14] In a specific sense that allows for error in stochastic descriptions.

Given the flaws in the conventional Macroeconomic paradigm, a broader objective of this research was not simply to improve on those models but to rather show how easy it can be to develop a very different macroeconomic paradigm. My hope is that this will lead other researchers to "...explore multiple models in a creative exploratory way", as Colander (2010) suggests in his institutional critique of Economics. This may describe an Economics community spread more thinly, but it is by spreading out that one finds something, not by all looking in the same place.

3.7.1 The other great question of economics

My new paradigm finds the origin of "boom and bust" not in instant economy-wide changes to technology, but rather in the breakdown of productive relationships due to enforcement failures. These failures are able to affect significant portions of the economy because they cause avalanches of failing enforcement, as enforcers who lose their own productive relationships lose the resources to enforce those between their enforcees. All this said, the volatility of the macroeconomy's boom and bust is not the only feature of interest in Macroeconomics. Adam Smith's seminal treatment of the macroeconomy, in book three of his great treatise, focusses on the growth in productivity of nations over time. In modern Economics growth is considered no less important, with Lucas (1988) famously saying of growth's welfare implications,

> Once one starts to think about them, it is hard to think about anything else.

These thoughts are echoed on the other side of the Neoclassical-Keynesian divide, with Mankiw (1995) stating that growth is "...perhaps more important – than short-run fluctuations".

Growth is an obvious feature of the model described in this paper, as the number of enforced relationships slowly builds over time. Why growth rates should differ

between countries is less well explained. Indeed, the massive and persistent disparity in GDP between different nations would seem to suggest a sensitivity to initial conditions more enduring than seen in this ergodic process. The extension I intend to pursue in subsequent work, that might reproduce such divergence requires the introduction of a second, competing, enforcement mechanism. While Hobbes's third party enforcement is a long standing theme in social philosophy, a more recently proposed mechanism comes in the *Social Capital* of Coleman (1988). This much abused term was original intended to describe the collateral value provided by the future value of relationships. In this way, short term conflicts of interest over contracts are enforced by the threat of losing the relationship with the other party and peers. I envisage this peer pressure based enforcement being described by the clustering in a network (the average number of common neighbours between each pair) but in conflict with the value of path lengths, which is an antagonistic property. This would let a single network model capture both mechanisms and the competition between them. However, in chapter 5 I give evidence that such a mechanism cannot be so easily described.

To sketch how competition between different enforcement mechanisms might lead to economic divergence, consider that with more peer-pressure based enforcement in place there is less incentive for individuals to subjugate themselves to third party enforcement. Hence, even if third party enforcement should become more effective with technological improvement, an initially high level of social capital might stop these gains in efficiency from being realised.

3.7.2 Policy implications

What are we to do with this new framework? What policies does it promote?

Traditionally macroeconomics has been obsessed with trying (or not trying) to control the economy by manipulating the money supply, either directly or through interest rates. To my knowledge there is no persistent statistical relationship between economic

aggregates and any measure of the money supply —but, of course, this would be the justification for a theoretical model linking the two. That said, there is no explicit treatment of money as a medium of exchange in DSGE models; it is merely a factor of the price level. The model here may similarly have seemed to ignore money and its actual role in exchange. But, this is not so. As fiat currencies rely on a legal mandate making them obligatory tender, they essentially act as a contract that must be enforced by the formal hierarchy. In this way money, and people's faith in a government enforcing the value of money, is actively considered in my model. That said, the precise numbers of coins and notes in circulation does not have an explicit link to the efficacy of the government's enforcement. Manipulating the money supply is therefore not policy tool in a Macroeconomics of Social Contracts, and it doesn't need to be.

That we can capture economic fluctuations without thinking about money or interest rates frees the central banker's hand to concern themselves with other issues, like solvency. Likewise, the unimportance of fiscal policy lets the government instead choose spending that will address social concerns arising from the coordination failure — redistribution to those who have lost jobs etc. More radically, an origin of economic fluctuations in enforcement relationships suggests changes to enforcement that might counter such problems. Specifically, any technology allowing more contracts to be enforced without a third party, would remove some of the dependence on a hierarchical structure that can be subject to avalanches.

One candidate is the social enforcement through peer pressure just discussed, the Social Capital of Coleman (1988). The power of community enforcement is limited to groups of about 150 people by our cognitive capacity to keep track of community structure, the so called Dunbar Number (Dunbar, 1992). To beat the Dunbar number, a technical solution would have to keep track of community structure for us, in a decentralised, secure, way that could not be abused by its administrator. This is a solution I pursue elsewhere.

Explicit enforcement need not be the only hierarchy that could be replaced by a techno-

logical alternative. Our society could soon be transformed by blind ballots conducted cheaply over the internet. Political parties in several Western democracies have begun to experiment with *Liquid Democracy* and *Crowd Source Governance*. Further, Iceland and Estonia have both seen internet voting actually deployed in local policy making. If these newly feasible delegative and direct democratic systems replaced representative democracy at a national level in large countries, then the global governance hierarchy would find itself significantly less hierarchical. A policy to stabilise the economy might then be to enable transitions to these systems of governance.

Although the model doesn't suggest a precise relationship between money supply and economic aggregates, unpredictable central banking could be speculated as undermining the credibility of currency guaranteeing exchange agreements. Hence, taking decisions over the money stock out of central bankers' hands could be seen as dismantling some of society's unstable enforcement hierarchy. The advent of the decentralised currency BitCoin[15] promises exactly this scenario.

A particularly important set of agreements that need enforcing are those ensuring the veracity of information passed around society. This can be surprisingly hierarchical, where an editor vouches for the quality of articles in a newspaper or academic journal and acts as an enforcer of the exchange. But, this position of information gatekeeper, puts such editors in a position to cause the cascades described by my model. Another recommendation system for quality information might therefore be preferable. In chapter 2 I parenthetically suggested that the role of academic journal editors might be replaced by topological metrics of the citation network. My suggestion is the Bonacich Centrality of the citation-coauthorship network. Like BitCoin, this system could be entirely decentralised: the desire to have a high centrality and be trusted would motivate authors of articles to log their references and coauthors, using encryption, and multiple records would secure against manipulation. So, another policy implication of

[15]BitCoin works through transparently keeping multiple independent accounts of every transaction that has ever taken place, so that the ownership of BitCoins (obligations to exchange with others) can be easily traced.

this model would be to develop a system along these lines to further dismantle social hierarchies.

3.7.3 Robustness: the power of indirect inference for long memory processes

This discussion of extensions and policy implications might be getting ahead of ourselves. The result in this chapter is some support for the hierarchical enforcement model as a description for macroeconomic fluctuations. But, more evidence is needed, particularly on the implications of the model for civil fluctuations, and the common parameters we would expect this to have with economic fluctuations. Even more important is the validity of the empirical method used here, and in the next chapter, I assess the power of Indirect Inference for a more conventional non-stationary model; I find support for its efficacy, that bolsters the empirical support found for my model in this chapter.

Bibliography

Acemoglu, D., Carvalho, V. M., Ozdaglar, A., and Tahbaz-Salehi, A. The network origins of aggregate fluctuations. *Econometrica*, 80(5):1977–2016, 09 2012. URL http://ideas.repec.org/a/ecm/emetrp/v80y2012i5p1977-2016.html.

Bak, P. *How nature works: the science of self-organized criticality*. Copernicus Series. Copernicus, 1996. ISBN 9780387947914. URL http://books.google.co.uk/books?id=e5XuAAAAMAAJ.

Bak, P., Chen, K., Scheinkman, J., and Woodford, M. Aggregate fluctuations from independent sectoral shocks: self-organized criticality in a model of production and inventory dynamics. *Ricerche Economiche*, 47(1):3–30, March 1993. URL http://ideas.repec.org/a/eee/riceco/v47y1993i1p3-30.html.

Binmore, K. *Game theory & the social contract*. MIT Press, 1998.

Binmore, K. Rational decisions in large worlds. *Annales d'Economie et de Statistique*, (86):25–41, 2007. URL http://ideas.repec.org/a/adr/anecst/y2007i86p04.html.

Colander, D. The economics profession, the financial crisis, and method. *Journal of Economic Methodology*, 17(4):419–427, 2010. URL http://ideas.repec.org/a/taf/jecmet/v17y2010i4p419-427.html.

Coleman, J. S. Social capital in the creation of human capital. *Amer. J. Sociol*, pages 95–120, 1988.

Dahlhaus, R. Efficient parameter estimation for self-similar processes. *The Annals of Statistics*, 17(4):pp. 1749–1766, 1989. ISSN 00905364. URL http://www.jstor.org/stable/2241663.

Dasgupta, P. Social capital and economic performance: Analytics. In editor, editor, *Foundations of Social Capital*. Edward Elgar, Cheltenham, UK, 2003.

Delli Gatti, D., Gallegati, M., Greenwald, B., Russo, A., and Stiglitz, J. E. The financial accelerator in an evolving credit network. *Journal of Economic Dynamics and Control*, 34(9):1627–1650, September 2010.

Demange, G. and Wooders, M. *Group formation in economics: networks, clubs and coalitions*. Cambridge University Press, 2005. ISBN 9780521842716.

Dopfer, K. Evolution and complexity in economics revisited. Papers on Economics and Evolution 2011-02, Max Planck Institute of Economics, Evolutionary Economics Group, Feb. 2011. URL http://ideas.repec.org/p/esi/evopap/2011-02.html.

Dunbar, R. Neocortex size as a constraint on group size in primates. *Journal of Human Evolution*, 22(6):469 – 493, 1992. ISSN 0047-2484. doi: 10.1016/0047-2484(92)90081-J. URL http://www.sciencedirect.com/science/article/pii/004724849290081J.

Finn, M. G. Variance properties of solow's productivity residual and their cyclical implications. *Journal of Economic Dynamics and Control*, 19(5-7):1249–1281, 1995. URL http://ideas.repec.org/a/eee/dyncon/v19y1995i5-7p1249-1281.html.

Gabaix, X. Power laws in economics and finance. *Annual Review of Economics*, 1(1):255–294, 05 2009. URL http://ideas.repec.org/a/anr/reveco/v1y2009p255-294.html.

Gaffeo, E., Gatti, D. D., Desiderio, S., and Gallegati, M. Adaptive microfoundations for emergent macroeconomics. *Eastern Economic Journal*, 34(4):441–463, 2008.

Gatti, D., Gaffeo, E., Gallegati, M., Giulioni, G., and Palestrini, A. *Emergent Macroeconomics: An Agent-Based Approach to Business Fluctuations*. New Economic Windows. Springer, 2008. ISBN 9788847007246. URL http://books.google.co.uk/books?id=ivHbzYGGdrkC.

Gatti, D., Delli, G., Desiderio, S., Gaffeo, E., Cirillo, P., and Gallegati, M. *Macroeconomics from the Bottom-up*. Springer, 2011. ISBN 9788847019706.

Gatti, D. D., Gallegati, M., Greenwald, B., Russo, A., and Stiglitz, J. E. Business fluctuations in a credit-network economy. *Physica A: Statistical Mechanics and its Applications*, 370(1):68 – 74, 2006.

Giocoli, N. Fixing the point: the contribution of early game theory to the tool-box of modern economics. *Journal of Economic Methodology*, 10(1):1–39, March 2003.

Grosskopf, B. and Nagel, R. The two-person beauty contest. *Games and Economic Behavior*, 62(1):93–99, January 2008. URL http://ideas.repec.org/a/eee/gamebe/v62y2008i1p93-99.html.

Harless, D. W. and Camerer, C. F. The predictive utility of generalized expected utility theories. *Econometrica*, 62(6):1251–89, November 1994. URL http://ideas.repec.org/a/ecm/emetrp/v62y1994i6p1251-89.html.

Hobbes, T. *Leviathan Or The Matter, Form, and Power of a Common-wealth, Ecclesiastical and Civil*. Andrew Crooke, 1651. URL http://books.google.co.uk/books?id=L3FgBpvIWRkC.

Jackson, M. *Social and Economic Networks*. Princeton University Press, 2010. ISBN 9780691148205.

Jackson, M. O. A survey of models of network formation: Stability and efficiency. Game theory and information, EconWPA, Mar. 2003. URL http://ideas.repec.org/p/wpa/wuwpga/0303011.html.

Jackson, M. O. The economics of social networks. In *PROCEEDINGS OF THE 9 TH WORLD CONGRESS OF THE ECONOMETRIC SOCIETY*, 2005.

Jackson, M. O. and Rogers, B. W. The economics of small worlds. *Journal of the European Economic Association*, 3(2-3):617–627, 04/05 2005a.

Jackson, M. O. and Rogers, B. W. The economics of small worlds. *Journal of the European Economic Association*, 3(2-3):617–627, 04/05 2005b.

Jackson, M. O., Rodriguez-Barraquer, T., and Tan, X. Social capital and social quilts: Network patterns of favor exchange. *American Economic Review*, 102(5):1857–97, August 2012. URL http://ideas.repec.org/a/aea/aecrev/v102y2012i5p1857-97.html.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

Kirman, A. *Complex Economics: Individual and Collective Rationality*. The Graz Schumpeter Lectures. Taylor & Francis Group, 2010. ISBN 9780415568555. URL http://books.google.co.uk/books?id=Lt55vIVpNLQC.

Le, V. P. M., Minford, P., and Wickens, M. The 'puzzles' methodology: En route to indirect inference? *Economic Modelling*, 27(6):1417–1428, November 2010. URL http://ideas.repec.org/a/eee/ecmode/v27y2010i6p1417-1428.html.

Le, V. P. M., Meenagh, D., Minford, P., and Wickens, M. How much nominal rigidity is there in the us economy? testing a new keynesian dsge model using indirect inference. *Journal of Economic Dynamics and Control*, 35(12):2078–2104, 2011. URL http://ideas.repec.org/a/eee/dyncon/v35y2011i12p2078-2104.html.

Le, V. P. M., Meenagh, D., Minford, P., and Wickens, M. Testing dsge models by indirect inference and other methods: some monte carlo experiments. Cardiff Economics Working Papers E2012/15, Cardiff University, Cardiff Business School, Econom-

ics Section, June 2012. URL http://ideas.repec.org/p/cdf/wpaper/2012-15.html.

Locke, J. *Two Treatises on Government*. R. Butler, 1821. URL http://books.google.co.uk/books?id=AM9qFIrSa7YC.

Lucas, R. J. On the mechanics of economic development. *Journal of Monetary Economics*, 22(1):3–42, July 1988. URL http://ideas.repec.org/a/eee/moneco/v22y1988i1p3-42.html.

Maddison, A. Historical statistics of the worlds economy: 1-2008 ad. URL http://www.ggdc.net/maddison/oriindex.htm.

Mandelbrot, B. and Van Ness, J. Fractional brownian motions, fractional noises and applications. *SIAM review*, 10(4):422–437, 1968.

Mankiw, G. The growth of nations. *Brookings Papers on Economic Activity*, 26(1, 25th A):275–326, 1995. URL http://ideas.repec.org/a/bin/bpeajo/v26y1995i1995-1p275-326.html.

Mayoral, L. Further evidence on the statistical properties of real gnp. Economics Working Papers 955, Department of Economics and Business, Universitat Pompeu Fabra, May 2005. URL http://ideas.repec.org/p/upf/upfgen/955.html.

Morris, I. and Farrar, S. *Why The West Rules - For Now: The Patterns of History and what they reveal about the Future*. Profile, 2010. ISBN 9781847652942. URL http://books.google.co.uk/books?id=bMi1XVXAl48C.

Rabin, M. Psychology and economics. *Journal of Economic Literature*, 36(1):pp. 11–46, 1998. ISSN 00220515. URL http://www.jstor.org/stable/2564950.

Shimotsu, K. Exact local whittle estimation of fractional integration with unknown mean and time trend. *Econometric Theory*, 26(02):501–540, April 2010.

Vega-Redondo, F. Building up social capital in a changing world. *Journal of Economic Dynamics and Control*, 30(11):2305–2338, November 2006.

Velasco, C. Non-stationary log-periodogram regression. *Journal of Econometrics*, 91(2):325–371, August 1999. URL http://ideas.repec.org/a/eee/econom/v91y1999i2p325-371.html.

Vespignani, A. and Zapperi, S. How self-organized criticality works: A unified mean-field picture. *Physical Review E*, 57(6):6345, 1998.

Wieland, V. and Wolters, M. The diversity of forecasts from macroeconomic models of the us economy. *Economic Theory*, 47(2):247–292, June 2011. URL http://ideas.repec.org/a/spr/joecth/v47y2011i2p247-292.html.

Young, H. *Strategic learning and its limits*. Oxford University Press, 2004. ISBN 9780199269181.

3.A Simulation Codes

All the models used in this chapter were implemented in Java, in order to give the greatest cross platform portability. Because Java is an object oriented language, the program is expressed in the form of several interrelated sub-programs, each keeping track of its own set of data. This functionality is intended to control the ways in which variables can be changed, by compartmentalising the program, to prevent unintended behaviour resulting from unintended access. But, for a program this size there is little danger, so I apologise to the reader for any difficulties they may have in reading object oriented code.

3.A.1 The standard hierarchical enforcement model

EnforceAgent.java

```
/*
 Copyright 2013 by Tom Wilkinson and Cardiff University
 Licensed under the Academic Free License version 3.0
 See the file "LICENSE" for more information
*/
package enforcers;
import java.io.*;
```

```
import util.*;

/* This class represents an agent which maintains enforcement
   relationships with
* other agents
*/
public class EnforceAgent implements Serializable
{
    int index;
    int numSubjects;
    Bag subjects;
    Bag domain;
    Bag superSovereigns;
    Bag component;

    public EnforceAgent(int index)
    {
        this.index = index;
        numSubjects = 0;
        subjects = new Bag();
        domain = new Bag();
        superSovereigns = new Bag();
    }
}
```

Hierarchy.java

```
/*
 Copyright 2013 by Tom Wilkinson and Cardiff University
 Licensed under the GNU General Public License version 3.0
 See the file "LICENSE" for more information
*/
package enforcers;
import util.*;

/**This class contains the methods which represent events
   affecting the
* enforcement hierarchy:
* - a network growth event
* - a network collapse event
* - a measuring of productive activity
*
* TERMINOLOGY (for sake of readability): enforcer = sovereign
   ; enforcee = subject;
```

```
* domain = all subjects of a sovereign, all those subjects'
    subjects etc ;
* superSovereigns = all those other agents who include the
    specific agent in
* their domains */
public class Hierarchy {
    int numAgents;
    int maxSubjects;
    double contagionProb;

    Bag freeMen;
    Bag components;

    MersenneTwisterFast random;
    Bag agents;
    int[] domainDistribution;

    /* we'll use a directed network, to capture the directed
       nature of
     * enforcement */
    public Hierarchy(int numAgents, int maxSubjects, double
       contagionProb){
        this.numAgents = numAgents;
        this.maxSubjects = maxSubjects;
        this.contagionProb = contagionProb;

        freeMen = new Bag();

        random = new MersenneTwisterFast(System.
           currentTimeMillis());
        agents = new Bag();
        domainDistribution = new int[numAgents];
    }

    /* Agents without sovereigns are offered a new sovereign */
    public void subjugate(){

        if(!freeMen.isEmpty())
        {
            /* A new node looks for an enforcer, chosen at
               random from those
             * without links */
            EnforceAgent newSubject
                    = (EnforceAgent)freeMen.objs[random.nextInt
                       (freeMen.numObjs)];
```

```
/* a potential enforcer is chosen at random from
   all nodes */
EnforceAgent sovereign =
            (EnforceAgent)agents.objs[random.
               nextInt(numAgents)];
/* there is guaranteed to be some sovereign out
   there who doesn't
 * have full degree, by virtue of the tree
    structure having leaves */
while(sovereign.equals(newSubject)||!(sovereign.
   numSubjects<maxSubjects)
        ||sovereign.superSovereigns.contains(
           newSubject)){
    sovereign = (EnforceAgent)agents.objs[random.
       nextInt(numAgents)];
}

sovereign.subjects.add(newSubject);
freeMen.remove(newSubject);
freeMen.remove(sovereign);

sovereign.numSubjects++;
/* The sovereign's domain, along with all their
   super-sovereigns'
 * domains, will now also include the newSubject
    and all their
 * existing domain */
sovereign.domain.add(newSubject);
sovereign.domain.addAll(newSubject.domain);
Bag superSovereigns = sovereign.superSovereigns;
int numSupers = superSovereigns.numObjs;
for(int superIndex=0; superIndex<numSupers;
   superIndex++){
    Bag superDomain = ((EnforceAgent)
       superSovereigns.objs[superIndex])
           .domain;
    superDomain.add(newSubject);
    superDomain.addAll(newSubject.domain);
}
/* meanwhile, the newSubject will now share all the
    sovereign's
 * super-sovereigns */
newSubject.superSovereigns.add(sovereign);
newSubject.superSovereigns.addAll(sovereign.
   superSovereigns);
/* As should every member of their domain! */
```

```
            Bag subDomain = newSubject.domain;
            int numSubjects = subDomain.size();
            for(int subjectIndex=0; subjectIndex<numSubjects;
                subjectIndex++){
                Bag subSuperSovereigns =
                        ((EnforceAgent)subDomain.objs[
                            subjectIndex])
                        .superSovereigns;
                if(!subSuperSovereigns.contains(sovereign))
                    subSuperSovereigns.add(sovereign);
                subSuperSovereigns.addAll(sovereign.
                    superSovereigns);
            }
        }
    }

    public void sovereignFail(){
        Bag failures = new Bag();

        /* randomly choose an agent to fail - though they may
           have no subjects */
        EnforceAgent initialFailure =
                (EnforceAgent)agents.objs[random.nextInt(
                    numAgents)];
        failures.add(initialFailure);

        /* all super sovereigns will lose this initialFailure's
            domain from
         * their own domains */
        Bag superSovereigns = initialFailure.superSovereigns;
        int numSupers = superSovereigns.numObjs;
        for(int superIndex=0; superIndex<numSupers; superIndex
            ++){
            ((EnforceAgent)superSovereigns.objs[superIndex]).
                domain
                    .removeAll(initialFailure.domain);
        }
        /* they lose their subjects and domain */
        initialFailure.numSubjects = 0;
        initialFailure.domain = new Bag();

        /* if the initialFailure had no supersovereigns, then
           they are a free
         * man again */
        if(superSovereigns.isEmpty()){
            if(!freeMen.contains(initialFailure)){
```

```
            freeMen.add(initialFailure);
        }
    }

    /* meanwhile, some sub-sovereigns lose everything and
       become freeMen
     * - I'll do this with a loop that constructs a Bag of
       each subsequent
     * tier, rather than by recursion that risks stack
       overflows (power law
     * of cascade sizes means some very big cascades, and
       deep recursions) */
    Bag unenforced = initialFailure.subjects;
    Bag lowerTierFail = new Bag();
    Bag lowerTierSafe = new Bag();
    while(unenforced.numObjs>0){
        EnforceAgent subject = (EnforceAgent)unenforced.
            objs[0];
        /* with a certain probability each subject will
            fail too */
        if(random.nextBoolean(contagionProb)){
            lowerTierFail.add(subject);
            failures.add(subject);
        }
        else lowerTierSafe.add(subject);
        /* regardless of whether the subject fails, the
            enforcement
         * relationships from those above will be lost */
        subject.superSovereigns = new Bag();
        unenforced.remove(subject);
    }
    while((lowerTierFail.numObjs>0) || (lowerTierSafe.
        numObjs>0)){
        Bag nextTierFail = new Bag();
        Bag nextTierSafe = new Bag();
        for(int subjectIndex=0;
                subjectIndex<lowerTierFail.numObjs;
                subjectIndex++){
            EnforceAgent sovereign
                    = (EnforceAgent)lowerTierFail.objs[
                        subjectIndex];
            /* this sovereign loses their subjects and
               domain */
            sovereign.numSubjects = 0;
            sovereign.domain = new Bag();
```

```
            /* ...returns to the set of agents without a
               sovereign (if they
             * weren't already)... */
            if(!freeMen.contains(sovereign)) freeMen.add(
               sovereign);
            else System.out.println("subject already free")
               ;
            /* ...and each of their subjects will suffer
               the same fate */
            unenforced = sovereign.subjects;
            while(unenforced.numObjs>0){
                EnforceAgent subject = (EnforceAgent)
                   unenforced.objs[0];
                /* with a given probability, this subject
                   fails too */
                if(random.nextBoolean(contagionProb)){
                    nextTierFail.add(subject);
                }
                else {
                    nextTierSafe.add(subject);
                }
                /* even if they do not fail, this subject
                   will still lose
                 * their relationships with the enforcers
                    above */
                subject.superSovereigns = new Bag();
                unenforced.remove(subject);
            }
        }
        for(int sovereignIndex=0;
            sovereignIndex<lowerTierSafe.numObjs;
            sovereignIndex++){
            EnforceAgent sovereign
                = (EnforceAgent)lowerTierSafe.objs[
                    sovereignIndex];
            /* Each of their subjects will also lose any
               sovereigns who have
             * failed above them
             * (the Bag passed here will also contain
                sovereigns who have
             * failed in different branches of the
                enforcement hierarchy,
             * but as they cannot be enforcers in any other
                 way there is no
             * harm passing them to the removeAll method */
            Bag lessEnforced = sovereign.subjects;
```

```
                for(int subjectIndx=0;
                        subjectIndx<lessEnforced.numObjs;
                        subjectIndx++){
                    EnforceAgent subject
                            = (EnforceAgent)lessEnforced.objs[
                                subjectIndx];
                    /* this subject's subjects will also lose
                       superSovereigns */
                    nextTierSafe.add(subject);
                    /* and this subject must lose those
                       supersovereigns that
                     * have failed */
                    subject.superSovereigns.removeAll(failures)
                        ;
                }
            }
            lowerTierFail = nextTierFail;
            lowerTierSafe = nextTierSafe;
        }
    }

    /* the formal economy will presumably be the largest
       component, output will
     * therefore be proportional to the number of pairs of
        agents in that
     * component */
    public double getOutput(){
        double output = 0;
        /* refresh the record of components, following growth
           and collapse*/
        discoverComponents();

        /* move through the components one by one, calculating
           the output and
         * keeping track of the highest; this will be the
            formal economy */
        int numComponents = components.numObjs;
        for(int componentIndex=0; componentIndex<numComponents;
            componentIndex++){
            int componentSubjects = ((Bag)components.objs[
                componentIndex])
                    .numObjs - 1;
            double componentOutput
                    = (double)componentSubjects*(double)(
                        componentSubjects-1);
            if(componentOutput>output)
```

```
                output = componentOutput;
        }
        if(output<0) System.err.println("negative output, in
            network object!");
        return output;
    }

    void discoverComponents(){
        /* refresh the bag of components */
        components = new Bag();

        /* any agent with no superSovereigns will be either
            their own component
         * or the sovereign to a component, so we need only
             explore the
         * components of such agents */
        for(int agentIndex=0; agentIndex<numAgents; agentIndex
            ++){
            EnforceAgent agent = (EnforceAgent)agents.objs[
                agentIndex];
            if(agent.superSovereigns.isEmpty())
            {
                Bag newComponent = new Bag(agent.domain);
                newComponent.add(agent);
                components.add(newComponent);
                int numMembers = newComponent.numObjs;
                for(int memberIndex=0; memberIndex<numMembers;
                    memberIndex++)
                {
                    ((EnforceAgent)newComponent.objs[
                        memberIndex]).component
                            = newComponent;
                }
            }
        }
    }

}
```

3.A.2 HierarchySim.java

```
/*
 Copyright 2013 by Tom Wilkinson and Cardiff University
 Licensed under the Academic Free License version 3.0
```

```
 See the file "LICENSE" for more information
*/
package enforcers;

import java.io.BufferedWriter;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.nio.charset.Charset;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.concurrent.*;
import util.FELW2St;

/** - Encapsulates and runs a hierarchical enforcement model
    for particular
*     parameter values
* - Samples the generated data at various frequencies, to see
     which best
*     reproduces the long-memory properties of the observed
    data
*/
public class HierarchySim implements Callable {
    final static Charset ENCODING = StandardCharsets.UTF_8;
    static final String OUTPUT_FILE_NAME = "sovs.csv";
    static final String SUMMARY_FILE_NAME = "summary.csv";
    static final String OUTPUT_PATH_NAME = "D:\\simulate\\sovs"
        ;

    int runVersion;
    double trueDiff;

    int numAgents;
    int simDuration;
    int numSims;
    int burnSize;
    int sampleSize;

    int maxSubjects;
    int failPeriod;
    double contagionProb;
```

```
Hierarchy society;

HierarchySim(int runVersion,double trueDiff,int numAgents,
   int simDuration,
        int numSims,int burnSize, int sampleSize,int
           maxSubjects,
        int failPeriod,double contagionProb){
    this.runVersion = runVersion;
    this.trueDiff = trueDiff;
    this.numAgents = numAgents;
    this.simDuration = simDuration;
    this.numSims = numSims;
    this.burnSize = burnSize;
    this.sampleSize = sampleSize;

    this.maxSubjects = maxSubjects;
    this.failPeriod = failPeriod;
    this.contagionProb = contagionProb;

    society = new Hierarchy(numAgents, maxSubjects,
       contagionProb);
}

public double[] call(){
    DateFormat dateFormat = new SimpleDateFormat("MMdd");
    Date date = new Date();
    String directory = OUTPUT_PATH_NAME
            +"_"+ numAgents +"_"+ runVersion +"_"+
               dateFormat.format(date)
            + "\\";
    new File(directory).mkdirs();
    /* a 2D array will hold the distributions of difference
       parameters for
     * the various sampling frequencies - but we need to
       work out what
     * frequencies are compatible with the sample size we
       want, IF sample
     * frequency is to double each iteration */
    int numFreq = 0;
    int minSample = sampleSize;
    while(minSample<=simDuration){
        minSample *= 2;
        numFreq++;
    }
    double[][] parDist = new double[numSims][numFreq];
```

```
for(int sim=0; sim<numSims; sim++){
    /* reset the Hierarchy object */
    society = new Hierarchy(numAgents, maxSubjects,
       contagionProb);
    for(int agent=0; agent<numAgents; agent++){
        newAgent = new EnforceAgent(agent)
        society.agents.add(newAgent);
        society.freeMen.add(newAgent);
    }
    /* the data is simulated for these parameters */
    double[] simData = new double[simDuration];
    int failTimer = failPeriod;
    for(int time=0; time<burnSize; time++){
        // new sovereign-subject relationship
        society.subjugate();

        // every failPeriod of time, a sovereign is
           selected to fail
        if(failTimer<1){
            society.sovereignFail();
            failTimer = failPeriod;
        }
        failTimer--;
        /* get output is run to refresh component sizes
            */
        society.getOutput();
    }
    for(int time=0; time<simDuration; time++){
        // new sovereign-subject relationship
        society.subjugate();

        // every failPeriod of time, a sovereign is
           selected to fail
        if(failTimer<1){
            society.sovereignFail();
            failTimer = failPeriod;
        }
        failTimer--;

        // finally, output is checked and recorded
        simData[time] = society.getOutput();
    }

    int sampleFreq = 1;
    for(int freqIndx=0; freqIndx<numFreq; freqIndx++){
        double[] sample = new double[sampleSize];
```

```
        int simPoint = simDuration - sampleFreq*
            sampleSize - 1;
        for(int dataPoint=0;
                dataPoint<sampleSize;
                dataPoint++){
            sample[dataPoint] = simData[simPoint];
            simPoint += sampleFreq;
        }

        parDist[sim][freqIndx] = new
                FELW2St(sample,(int)Math.floor(Math.pow
                    (sampleSize,0.65)),3)
                .estimate();
        sampleFreq *= 2;
    }

    /* backup this expensive simData in case of later
       need */
    try{
        // the output file needs to be named so as to
           be uniquely identified
        // (hence the date), but give relevant
           information quickly (hence params)
        Path outputPath = Paths.get(directory +
           maxSubjects +"_"+
                failPeriod +"_"+ contagionProb +"_"+
                    sim + OUTPUT_FILE_NAME );
        File outputFile = new File(outputPath.toString
           ());
        outputFile.createNewFile();

        try(BufferedWriter writer = Files.
           newBufferedWriter(outputPath,
                ENCODING)){
            // Let's print out the whole parameter list
            writer.write("Enforcement␣model␣parameters:
               ␣maxSubjects␣=␣"
                    + maxSubjects + ",␣failPeriod␣=␣" +
                        failPeriod
                    + ",␣contagionProb␣=␣" +
                        contagionProb);
            writer.newLine();
            writer.write("Simulated␣GDP...");
            writer.newLine();
            for(int time=0; time<simDuration; time++){
```

```
                    writer.write(Double.toString(simData[
                        time]));
                    writer.newLine();
                }
            }
        }
        catch (FileNotFoundException e)
        {
            System.err.println("FileNotFoundException: " +
                e.getMessage());
        }
        catch (IOException e)
        {
            System.err.println("Caught IOException: " + e.
                getMessage());
        }
    }

    /* find the sampleFrequency that gets the mean diffPar
        closest to
     * trueDiff */
    double bestDiffPar = 0.0;
    double bestDiffParVar = 0.0;
    double bestPVal = 0.0;
    double[] bestWaldDist = new double[numSims];
    int freq = 1;
    int bestFreq = 1;
    int bestFreqIndx = 1;
    double[] diffParDist = new double[numFreq];
    double[] diffParVarDist = new double[numFreq];
    for(int freqIndx=0; freqIndx<numFreq; freqIndx++){
        double meanDiffPar = 0.0;
        double varDiffPar = 0.0;
        for(int sim=0; sim<numSims; sim++){
            double diffPar = parDist[sim][freqIndx];
            meanDiffPar += diffPar;
            varDiffPar += diffPar*diffPar;
        }
        meanDiffPar /= numSims;
        diffParDist[freqIndx] = meanDiffPar;
        varDiffPar /= numSims;
        varDiffPar -= meanDiffPar*meanDiffPar;
        diffParVarDist[freqIndx] = varDiffPar;
        /* we now produce the Wald distribution, and count
            the p-value for
```

```
         * the hypothesis that the data generating
             mechanism has the same
         * long memory properties as the theoretical model
             */
        double[] waldDist = new double[numSims];
        double trueWald = (trueDiff - meanDiffPar);
        trueWald *= trueWald;
        trueWald /= varDiffPar;
        /* sequentially compare wald values with true value
            and track how many
         * are more extreme */
        double pVal = 0.0;
        for(int sim=0; sim<numSims; sim++){
            double waldVal = (parDist[sim][bestFreqIndx] -
                meanDiffPar);
            waldVal *= waldVal;
            waldVal /= varDiffPar;
            waldDist[sim] = waldVal;
            if(waldVal > trueWald){
                pVal += 1.0;
            }
        }
        pVal /= (double)numSims;
        if(pVal>bestPVal){
            bestDiffPar = meanDiffPar;
            bestDiffParVar = varDiffPar;
            bestWaldDist = waldDist;
            bestPVal = pVal;
            bestFreq = freq;
            bestFreqIndx = freqIndx;
        }
        freq *= 2;
    }

    /* record the best frequency and difference parameter
       for these pars */
    try{
        // the output file needs to be named so as to be
           uniquely identified
        // (hence the date), but give relevant information
           quickly (hence params)
        Path outputPath = Paths.get(directory + maxSubjects
            +"_"+
                failPeriod +"_"+ contagionProb +
                    SUMMARY_FILE_NAME );
        File outputFile = new File(outputPath.toString());
```

```
outputFile.createNewFile();

try(BufferedWriter writer = Files.newBufferedWriter
   (outputPath,
        ENCODING)){
    // Let's print out the whole parameter list
    writer.write("Enforcement␣model␣parameters:␣
       maxSubjects␣=␣"
            + maxSubjects + ",␣failPeriod␣=␣" +
               failPeriod
            + ",␣contagionProb␣=␣" + contagionProb)
               ;
    writer.newLine();
    writer.write("Implementation␣parameters:␣
       numAgents␣=␣"
            + numAgents + ",␣burnSize␣=␣" +
               burnSize
            + ",␣sampleSize␣=␣" + sampleSize + ",␣
               numSims␣=␣"
            + numSims + ",␣trueDiff␣=␣" + trueDiff)
               ;
    writer.newLine();
    writer.write("Best␣fractional␣difference␣
       parameter␣=␣"
            + bestDiffPar);
    writer.newLine();
    writer.write("Variance␣for␣best␣frequency␣=␣" +
        bestDiffParVar);
    writer.newLine();
    writer.write("Best␣frequency␣=␣" + bestFreq);
    writer.newLine();
    writer.write("p-Value␣for␣data␣based␣Wald␣value
       ␣=␣" + bestPVal);
    writer.newLine();
    writer.write("Other␣frequencies:␣");
    freq = 1;
    for(int freqIndx=0; freqIndx<numFreq; freqIndx
       ++){
     writer.newLine();
     writer.write(freq +",␣"+ diffParDist[
        freqIndx]
             +",␣"+ diffParVarDist[freqIndx]);
     freq *= 2;
    }
    writer.newLine();
    writer.write("best␣wald␣distribution:␣");
```

```
                for(int sim=0; sim<numSims; sim++){
                    writer.newLine();
                    writer.write(Double.toString(bestWaldDist[
                        sim]));
                }
            }
        }
        catch (FileNotFoundException e)
        {
            System.err.println("FileNotFoundException: " + e.
                getMessage());
        }
        catch (IOException e)
        {
            System.err.println("Caught IOException: " + e.
                getMessage());
        }

        double[] output = new double[2];
        output[0] = bestDiffPar;
        output[1] = bestPVal;
        return output;
    }
}
```

ThirdParties.java

```
/*
 Copyright 2013 by Tom Wilkinson and Cardiff University
 Licensed under the Academic Free License version 3.0
 See the file "LICENSE" for more information
*/
package enforcers;
import java.io.*;
import java.text.*;
import java.util.*;
import util.MersenneTwisterFast;


/* Use simulated annealing to find the parameter choices that
    best match the
* long memory properties of US GNP with those of a growing and
    collapsing
* enforcement network
*/
```

```
public class ThirdParties {
   // the output file needs to be named so as to be uniquely
      identified
   // (hence the date), but give relevant information quickly
      (hence params)
   DateFormat dateFormat = new SimpleDateFormat("MMdd");
   Date date = new Date();


   String outputFileName = "svrns.txt";
   String outputPathName = "C:\\simulate\\sovsaw";
   File outputFile;
   File outputPath;
   PrintWriter out;

   static double defaultTrueDiff = 0.8456;

   static int defaultRunVersion = 1;
   static int defaultNumAgents = 10000;

   static int defaultMaxSubjects = 5;
   static int defaultFailPeriod = 3;
   static double defaultContagionProb = 0.95;

   static int defaultInovSubjects = 1;
   static int defaultInovPeriod = 1;
   static double defaultInovProb = 0.01;

   static int defaultLBoundSubjects = 1;
   static int defaultLBoundPeriod = 1;
   static double defaultLBoundProb = 0.01;

   static int defaultUBoundSubjects = 20;
   static int defaultUBoundPeriod = 50;
   static double defaultUBoundProb = 1.00;

   static int defaultTimeBudget = 1000;
   static double defaultMinPVal = 0.95;
//    static double defaultMaxDiscrepancy = 0.01;
   static double defaultInitialTemp = 1000.0;

   static int defaultDuration = 10000;
   static int defaultBurnSize = 10000;
   static int defaultSampleSize = 150;
   static int defaultNumSims = 1000;
```

```
public static void main(String[] args){
    // print help
    if (keyExists("-help", args, 0)){
        System.err.println(
            "Format:␣␣␣␣␣␣␣␣␣␣␣java␣-jar␣mason.jar␣[-
                numAgents␣n]␣...␣\n\n" +
            "-help␣␣␣␣␣␣␣␣␣␣␣␣Shows␣this␣message␣and␣exits
                .\n\n" +
            "-runVersion␣␣␣␣␣␣␣int␣>␣1:␣a␣designation␣for␣
                this␣thread␣\n\n" +
            "-numAgents␣␣␣␣␣␣␣␣int␣>␣1:␣the␣number␣of␣
                agents␣in␣the␣simulation␣\n\n" +
            "-trueDiff␣␣␣␣␣␣␣␣double:␣the␣target␣
                difference␣parameter␣value␣\n\n" +
            "-maxSubjects␣␣␣␣␣␣int␣>␣1:␣the␣maximum␣number␣
                of␣subjects␣allowed␣\n\n" +
            "-failPeriod␣␣␣␣␣␣␣int␣>␣1:␣the␣number␣of␣
                network␣growth␣events␣between\n"
          + "␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣each␣network␣collapse␣event␣
            \n\n" +
            "-contagionProb␣␣␣␣0<␣double␣<1:␣the␣
                probability␣with␣which␣each␣subject\n"
          + "␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣themselves␣fail\n\n" +
            "-inovSubjects␣␣␣␣␣int␣>␣1:␣the␣maximum␣number␣
                of␣subjects␣allowed␣\n\n" +
            "-inovPeriod␣␣␣␣␣␣␣int␣>␣1:␣the␣number␣of␣
                network␣growth␣events␣between\n"
          + "␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣each␣network␣collapse␣event␣
            \n\n" +
            "-inovProb␣␣␣␣␣␣␣␣0<␣double␣<1:␣the␣
                probability␣with␣which␣each␣subject\n"
          + "␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣themselves␣fail\n\n" +
            "-lBoundSubjects␣␣␣int␣>␣1:␣the␣maximum␣number␣
                of␣subjects␣allowed␣\n\n" +
            "-lBoundPeriod␣␣␣␣␣int␣>␣1:␣the␣number␣of␣
                network␣growth␣events␣between\n"
          + "␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣each␣network␣collapse␣event␣
            \n\n" +
            "-lBoundProb␣␣␣␣␣␣␣0<␣double␣<1:␣the␣
                probability␣with␣which␣each␣subject\n"
          + "␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣themselves␣fail\n\n" +
            "-uBoundSubjects␣␣␣int␣>␣1:␣the␣maximum␣number␣
                of␣subjects␣allowed␣\n\n" +
            "-uBoundPeriod␣␣␣␣␣int␣>␣1:␣the␣number␣of␣
                network␣growth␣events␣between\n"
```

```
        + "                    each network collapse event
            \n\n" +
          "-uBoundProb       0< double <1: the
              probability with which each subject\n"
        + "                    themselves fail\n\n" +
          "-timeBudget       0< int: the convergence
              criterion \n\n" +
          "-maxDiscrepancy   0< double <1: the
              probability with which each subject
              themselves fail\n\n" +
          "-simDuration      int > 0: the simulation must
              stop when N\n" +
          "                  simulation steps have
              transpired.\n\n" +
          "-initialTemp      int > 0: the \"temperature\"
              for simulated annealing.\n\n" +
          "-burnSize         int > 0: the number of sim
              steps to ignore.\n\n" +
          "-sampleSize       int > 0: number of
              observations on which to \n" +
          "                  estimate the difference
              parameter.\n\n" +
          "-numSims          int > 0: number of
              simulations for some pars.\n\n");
      System.exit(0);
      }

java.text.NumberFormat n = java.text.NumberFormat.
    getInstance();
n.setMinimumFractionDigits(0);

// set trueParameter objective
double trueDiff = defaultTrueDiff;
String trueDiff_s = argumentForKey("-trueDiff", args,
    0);
if (trueDiff_s != null)
    try {
        trueDiff = Double.parseDouble(trueDiff_s);
        }
    catch (Exception e){
        throw new RuntimeException("Invalid 'trueDiff'
            value: "
                + trueDiff_s + ", must be in unit
                    interval");
        }
```

```
// set runVersion
int runVersion = defaultRunVersion;
String runVersion_s = argumentForKey("-runVersion",
   args, 0);
if (runVersion_s != null)
    try{
        runVersion = Integer.parseInt(runVersion_s);
        if (runVersion < 2) throw new Exception();
        }
    catch (Exception e){
        throw new RuntimeException("Invalid␣'runVersion
            '␣value:␣"
                + runVersion_s + ",␣must␣be␣greater␣
                    than␣1");
        }

// set numAgents
int numAgents = defaultNumAgents;
String numAgents_s = argumentForKey("-numAgents", args,
    0);
if (numAgents_s != null)
    try{
        numAgents = Integer.parseInt(numAgents_s);
        if (numAgents < 2) throw new Exception();
        }
    catch (Exception e){
        throw new RuntimeException("Invalid␣'numAgents'
            ␣value:␣"
                + numAgents_s + ",␣must␣be␣greater␣than
                    ␣1");
        }

// set maxSubjects
int maxSubjects = defaultMaxSubjects;
String maxSubjects_s = argumentForKey("-maxSubjects",
   args, 0);
if (maxSubjects_s != null)
    try {
        maxSubjects = Integer.parseInt(maxSubjects_s);
        if (maxSubjects < 2) throw new Exception();
        }
    catch (Exception e){
        throw new RuntimeException("Invalid␣'
            maxSubjects'␣value:␣"
                + maxSubjects_s + ",␣must␣be␣greater␣
                    than␣1");
```

```
        }

    // set failure period
    int failPeriod = defaultFailPeriod;
    String failPeriod_s = argumentForKey("-failPeriod",
       args, 0);
    if (failPeriod_s != null)
        try {
            failPeriod = Integer.parseInt(failPeriod_s);
            if (failPeriod < 1)
                throw new Exception();
            }
        catch (Exception e){
            throw new RuntimeException("Invalid␣'failPeriod
               '␣value:␣"
                    + failPeriod_s + ",␣must␣be␣greater␣
                       than␣1");
            }

    // set contagion probability
    double contagionProb = defaultContagionProb;
    String contagionProb_s = argumentForKey("-contagionProb
       ", args, 0);
    if (contagionProb_s != null)
        try {
            contagionProb = Double.parseDouble(
               contagionProb_s);
            if (contagionProb > 1.0 || contagionProb < 0.0)
                throw new Exception();
            }
        catch (Exception e){
            throw new RuntimeException("Invalid␣'
               contagionProb'␣value:␣"
                    + contagionProb_s + ",␣must␣be␣in␣unit␣
                       interval");
            }

    // set lBoundSubjects
    int lBoundSubjects = defaultLBoundSubjects;
    String lBoundSubjects_s = argumentForKey("-
       lBoundSubjects", args, 0);
    if (lBoundSubjects_s != null)
        try {
            lBoundSubjects = Integer.parseInt(
               lBoundSubjects_s);
            if (lBoundSubjects < 2) throw new Exception();
```

```
            }
        catch (Exception e){
            throw new RuntimeException("Invalid␣'
                lBoundSubjects'␣value:␣"
                    + lBoundSubjects_s + ",␣must␣be␣greater
                        ␣than␣1");
            }

    // set lBoundure period
    int lBoundPeriod = defaultLBoundPeriod;
    String lBoundPeriod_s = argumentForKey("-lBoundPeriod",
        args, 0);
    if (lBoundPeriod_s != null)
        try {
            lBoundPeriod = Integer.parseInt(lBoundPeriod_s)
                ;
            if (lBoundPeriod < 1)
                throw new Exception();
            }
        catch (Exception e){
            throw new RuntimeException("Invalid␣'
                lBoundPeriod'␣value:␣"
                    + lBoundPeriod_s + ",␣must␣be␣greater␣
                        than␣1");
            }

    // set lBound probability
    double lBoundProb = defaultLBoundProb;
    String lBoundProb_s = argumentForKey("-lBoundProb",
       args, 0);
    if (lBoundProb_s != null)
        try {
            lBoundProb = Double.parseDouble(lBoundProb_s);
            if (lBoundProb > 1.0 || lBoundProb < 0.0)
                throw new Exception();
            }
        catch (Exception e){
            throw new RuntimeException("Invalid␣'lBoundProb
                '␣value:␣"
                    + lBoundProb_s + ",␣must␣be␣in␣unit␣
                        interval");
            }

    // set uBoundSubjects
    int uBoundSubjects = defaultUBoundSubjects;
```

```
String uBoundSubjects_s = argumentForKey("-
    uBoundSubjects", args, 0);
if (uBoundSubjects_s != null)
    try {
        uBoundSubjects = Integer.parseInt(
            uBoundSubjects_s);
        if (uBoundSubjects < 2) throw new Exception();
        }
    catch (Exception e){
        throw new RuntimeException("Invalid␣'
            uBoundSubjects'␣value:␣"
                + uBoundSubjects_s + ",␣must␣be␣greater
                    ␣than␣1");
        }

// set uBoundure period
int uBoundPeriod = defaultUBoundPeriod;
String uBoundPeriod_s = argumentForKey("-uBoundPeriod",
    args, 0);
if (uBoundPeriod_s != null)
    try {
        uBoundPeriod = Integer.parseInt(uBoundPeriod_s)
            ;
        if (uBoundPeriod < 1)
            throw new Exception();
        }
    catch (Exception e){
        throw new RuntimeException("Invalid␣'
            uBoundPeriod'␣value:␣"
                + uBoundPeriod_s + ",␣must␣be␣greater␣
                    than␣1");
        }

// set uBound probability
double uBoundProb = defaultUBoundProb;
String uBoundProb_s = argumentForKey("-uBoundProb",
    args, 0);
if (uBoundProb_s != null)
    try {
        uBoundProb = Double.parseDouble(uBoundProb_s);
        if (uBoundProb > 1.0 || uBoundProb < 0.0)
            throw new Exception();
        }
    catch (Exception e){
        throw new RuntimeException("Invalid␣'uBoundProb
            '␣value:␣"
```

```
                + uBoundProb_s + ", must be in unit
                    interval");
            }

        // set inovSubjects
        int inovSubjects = defaultInovSubjects;
        String inovSubjects_s = argumentForKey("-inovSubjects",
            args, 0);
        if (inovSubjects_s != null)
            try {
                inovSubjects = Integer.parseInt(inovSubjects_s)
                    ;
                if (inovSubjects < 2) throw new Exception();
                }
            catch (Exception e){
                throw new RuntimeException("Invalid '
                    inovSubjects' value: "
                        + inovSubjects_s + ", must be greater
                            than 1");
                }

        // set inovure period
        int inovPeriod = defaultInovPeriod;
        String inovPeriod_s = argumentForKey("-inovPeriod",
            args, 0);
        if (inovPeriod_s != null)
            try {
                inovPeriod = Integer.parseInt(inovPeriod_s);
                if (inovPeriod < 1)
                    throw new Exception();
                }
            catch (Exception e){
                throw new RuntimeException("Invalid 'inovPeriod
                    ' value: "
                        + inovPeriod_s + ", must be greater
                            than 1");
                }

        // set inov probability
        double inovProb = defaultInovProb;
        String inovProb_s = argumentForKey("-inovProb", args,
            0);
        if (inovProb_s != null)
            try {
                inovProb = Double.parseDouble(inovProb_s);
                if (inovProb > 1.0 || inovProb < 0.0)
```

```
                throw new Exception();
            }
        catch (Exception e){
            throw new RuntimeException("Invalid␣'inovProb'␣
                value:␣"
                    + inovProb_s + ",␣must␣be␣in␣unit␣
                        interval");
            }

    // set the convergence criterion
    double minPVal = defaultMinPVal;
    String minPVal_s = argumentForKey("-minPVal", args, 0);
    if (minPVal_s != null)
        try {
            minPVal = Double.parseDouble(minPVal_s);
            if (minPVal > 1.0 || minPVal < 0.0)
                throw new Exception();
            }
        catch (Exception e){
            throw new RuntimeException("Invalid␣'minPVal'␣
                value:␣"
                    + minPVal_s + ",␣must␣be␣in␣unit␣
                        interval");
            }

    // set the time budget
    int timeBudget = defaultTimeBudget;
    String timeBudget_s = argumentForKey("-timeBudget",
        args, 0);
    if (timeBudget_s != null)
        try {
            timeBudget = Integer.parseInt(timeBudget_s);
            if (timeBudget < 1)
                throw new Exception();
            }
        catch (Exception e){
            throw new RuntimeException("Invalid␣'timeBudget
                '␣value:␣"
                    + timeBudget_s + ",␣must␣be␣in␣unit␣
                        interval");
            }

    int simDuration = defaultDuration;
    String simDuration_s = argumentForKey("-simDuration",
        args, 0);
    if (simDuration_s != null)
```

```
        try{
            simDuration = Integer.parseInt(simDuration_s);
            if (simDuration < 0) throw new Exception();
            }
        catch (Exception e){
            throw new RuntimeException("Invalid ' 
                simDuration' value: " + simDuration_s + ", 
                must be an integer >= 0");
            }

    double initialTemp = defaultInitialTemp;
    String initialTemp_s = argumentForKey("-initialTemp",
        args, 0);
    if (initialTemp_s != null)
        try{
            initialTemp = Double.parseDouble(initialTemp_s)
                ;
            if (initialTemp < 0) throw new Exception();
            }
        catch (Exception e){
            throw new RuntimeException("Invalid ' 
                initialTemp' value: " + initialTemp_s + ", 
                must be a double >= 0");
            }

    int burnSize = defaultBurnSize;
    String burnSize_s = argumentForKey("-burnSize", args,
        0);
    if (burnSize_s != null)
        try{
            burnSize = Integer.parseInt(burnSize_s);
            if (burnSize < 0) throw new Exception();
            }
        catch (Exception e){
            throw new RuntimeException("Invalid 'burnSize' 
                value: " + burnSize_s + ", must be an 
                integer >= 0");
            }

    int sampleSize = defaultSampleSize;
    String sampleSize_s = argumentForKey("-sampleSize",
        args, 0);
    if (sampleSize_s != null)
        try{
            sampleSize = Integer.parseInt(sampleSize_s);
            if (sampleSize < 0) throw new Exception();
```

```
            }
        catch (Exception e){
            throw new RuntimeException("Invalid 'sampleSize
                ' value: " + sampleSize_s + ", must be an
                integer >= 0");
            }

    int numSims = defaultNumSims;
    String numSims_s = argumentForKey("-numSims", args, 0);
    if (numSims_s != null)
        try{
            numSims = Integer.parseInt(numSims_s);
            if (numSims < 0) throw new Exception();
            }
        catch (Exception e){
            throw new RuntimeException("Invalid 'numSims'
                value: " + numSims_s + ", must be an integer
                >= 0");
            }

    /* now for simulated annealing:
     * - run simulations for a random set of pars
        neighbouring the initial ones
     * - try all the different sample frequencies to find
        the best
     * - move to another state with probability given by
        the free energy of
     *   the system
     * - record the state of the system and the pars in
        case of crash */
    double bestDiffPar = 0.0;
    int bestMaxSubjects = maxSubjects;
    int bestFailPeriod = failPeriod;
    double bestContagionProb = contagionProb;

    int iterations = 1;
    double temperature = initialTemp;
    double pVal = 0.0;
    MersenneTwisterFast random = new MersenneTwisterFast();
    while(pVal<minPVal && iterations<timeBudget){
        /* choose a random neighbour - one of the variables
            to peterb */
        int newMaxSubjects = maxSubjects;
        int newFailPeriod = failPeriod;
        double newContagionProb = contagionProb;
        double varSelect = random.nextDouble();
```

```
/* uniform probability of one of six moves, derived
    by dividing the
 * unit interval into six: */
double boundary = 1.0/6.0; // cheaper to multiply
   than divide later
if(varSelect < boundary){
    if(maxSubjects > lBoundSubjects){
        newMaxSubjects -= inovSubjects;
    }
    else {
        newMaxSubjects += inovSubjects;
    }
}
else if(varSelect < 2.0*boundary){
    if(maxSubjects < uBoundSubjects){
        newMaxSubjects += inovSubjects;
    }
    else{
        newMaxSubjects -= inovSubjects;
    }
}
else if(varSelect < 3.0*boundary){
    if(failPeriod > lBoundPeriod){
        newFailPeriod -= inovPeriod;
    }
    else{
        newFailPeriod += inovPeriod;
    }
}
else if(varSelect < 4.0*boundary){
    if(failPeriod < uBoundPeriod){
        newFailPeriod += inovPeriod;
    }
    else {
        newFailPeriod -= inovPeriod;
    }
}
else if(varSelect < 5.0*boundary){
    if(contagionProb > lBoundProb){
        newContagionProb -= inovProb;
    }
    else {
        newContagionProb += inovProb;
    }
}
else {
```

```
            if(contagionProb < uBoundProb){
                newContagionProb += inovProb;
            }
            else {
                newContagionProb -= inovProb;
            }
        }

        /* simulate the neighbour */
        double[] output = new HierarchySim(runVersion,
            trueDiff, numAgents,
                simDuration, numSims, burnSize, sampleSize,
                    newMaxSubjects,
                newFailPeriod, newContagionProb).call();
        double neighbourDiff = output[0];
        double newPVal = output[1];
        double improvement = pVal - newPVal;
        if(improvement < 0){
            /* this neighbour improves on the previous pars
                , then we move */
            maxSubjects = newMaxSubjects;
            failPeriod = newFailPeriod;
            contagionProb = newContagionProb;
            pVal = newPVal;
            /* and we also update the global bests */
            bestMaxSubjects = newMaxSubjects;
            bestFailPeriod = newFailPeriod;
            bestContagionProb = newContagionProb;
            bestDiffPar = neighbourDiff;
        }
        /* if not, we still move with a probability that
            diminishes with
         * both system "temperature" and the deterioration
            */
        else if(random.nextDouble() < Math.exp(-improvement
            /temperature)){
            maxSubjects = newMaxSubjects;
            failPeriod = newFailPeriod;
            contagionProb = newContagionProb;
            pVal = newPVal;
        }

        iterations++;
        temperature = initialTemp/(double)iterations; //
            Cauchy annealing schedule
    }
```

```
        /* output the best pars we've seen */
        System.out.printf("Optimal pars: maxSubjects=" +
            bestMaxSubjects +
                ", failPeriod=" + bestFailPeriod +
                ", contagionProb=" + bestContagionProb + "\n\n"
                    +
                "Optimal p-value=" + pVal + "\n" +
                "Optimal difference parameter=" + bestDiffPar);
        System.exit(0);
    }

    static String argumentForKey(String key, String[] args, int
        startingAt){
        for(int x=0;x<args.length-1;x++)  // key can't be the
            last string
             if (args[x].equalsIgnoreCase(key))
                 return args[x + 1];
        return null;
        }

    static boolean keyExists(String key, String[] args, int
       startingAt){
        for(int x=0;x<args.length;x++)  // key can't be the
            last string
             if (args[x].equalsIgnoreCase(key))
                 return true;
        return false;
        }
}
```

3.A.3 Utilities used in this simulation

Bag.java

```
/*
 Copyright 2006 by Sean Luke and George Mason University
 Licensed under the Academic Free License version 3.0
 See the file "LICENSE" for more information
*/

package util;
import java.util.*;
import java.lang.reflect.*;
```

```
/** Maintains a simple array (objs) of Objects and the number
    of objects (numObjs) in the array
    (the array can be bigger than this number).  Unlike Vector
       or ArrayList, Bag is designed
    to encourage direct access of the array.  If you access the
        objects directly, they are
    stored in positions [0 ... numObjs-1].  If you wish to
       extend the array, you should call
    the resize method.

    <p>By providing direct access to the array, Bags are about
       three and a half times faster than ArrayLists
    (whose get/set methods unfortunately at present contain un-
       inlinable range bounds checks) and four times faster
    than Vectors (whose methods additionally are synchronized).
        Even Bag's built-in get() and set() methods,
    complete with range bounds checks, are twice the speed of
       ArrayLists.  To get faster
    than a Bag, you'd have to go to a raw fixed-length array of
        the specific class type of your objects.
    Accessing a Bag's Object array and casting its Objects into
        the appropriate class is about 50% slower
    than accessing a fixed-length array of that class in the
       first place.

    <p>Bag is not synchronized, and so should not be accessed
       from different threads without locking on it
    or some appropriate lock object first.  Bag also has an
       unusual, fast method for removing objects
    called remove(...), which removes the object simply by
       swapping the topmost object into its
    place.  This means that after remove(...) is called, the
       Bag may no longer have the same order
    (hence the reason it's called a "Bag" rather than some
       variant on "Vector" or "Array" or "List").  You can
    guarantee order by calling removeNondestructively(...)
       instead if you wish, but this is O(n) in the worst case.

    <p>Bags provide iterators but you are strongly encouraged
       to just access the array instead.  Iterators
    are slow.  Bag's iterator performs its remove operation by
       calling removeNondestructively().
    Like array access, iterator usage is undefined if objects
       are placed into the Bag or
```

```
   removed from the Bag in the middle of the iterator usage (
      except by using the iterator's remove
   operation of course).
*/

public class Bag implements java.util.Collection, java.io.
   Serializable, Cloneable, Indexed
   {
   public Object[] objs;
   public int numObjs;

   public Bag() { numObjs = 0; objs = new Object[1]; }

   /** Creates a Bag with a given initial capacity. */
   public Bag(int capacity) { numObjs = 0; objs = new Object[
      capacity]; }

   /** Adds the objects from the other Bag without copying
      them.  The size of the
       new Bag is the minimum necessary size to hold the
          objects. */
   public Bag(final Bag other)
       {
       if (other==null) { numObjs = 0; objs = new Object[1]; }
       else
           {
           numObjs = other.numObjs;
           objs = new Object[numObjs];
           System.arraycopy(other.objs,0,objs,0,numObjs);
           }
       }

   public int size()
       {
       return numObjs;
       }

   public boolean isEmpty()
       {
       return (numObjs<= 0);
       }

   public boolean addAll(final Collection other)
       {
       if (other instanceof Bag) return addAll((Bag)other);
          // avoid an array build
```

```
    return addAll(numObjs, other.toArray());
    }

  public boolean addAll(final int index, final Collection
     other)
    {
    if (other instanceof Bag) return addAll(index, (Bag)
       other);  // avoid an array build
    return addAll(index, other.toArray());
    }

  public boolean addAll(final int index, final Object[] other
     )
    {
    // throws NullPointerException if other == null,
    // ArrayIndexOutOfBoundsException if index < 0,
    // and IndexOutOfBoundsException if index > numObjs
    if (index > numObjs) { throwIndexOutOfBoundsException(
       index); }
    if (other.length == 0) return false;
    // make Bag big enough
    if (numObjs+other.length > objs.length)
        resize(numObjs+other.length);
    if (index != numObjs)    // make room
        System.arraycopy(objs,index,objs,index+other.length
           ,other.length);
    System.arraycopy(other,0,objs,index,other.length);
    numObjs += other.length;
    return true;
    }

  public boolean addAll(final Bag other) { return addAll(
     numObjs,other); }

  public boolean addAll(final int index, final Bag other)
    {
    // throws NullPointerException if other == null,
    // ArrayIndexOutOfBoundsException if index < 0,
    // and IndexOutOfBoundsException if index > numObjs
    if (index > numObjs) { throwIndexOutOfBoundsException(
       index); }
    if (other.numObjs <= 0) return false;
    // make Bag big enough
    if (numObjs+other.numObjs > objs.length)
        resize(numObjs+other.numObjs);
    if (index != numObjs)     // make room
```

```
            System.arraycopy(objs,index,objs,index+other.
                numObjs,other.numObjs);
        System.arraycopy(other.objs,0,objs,index,other.numObjs)
            ;
        numObjs += other.numObjs;
        return true;
        }

    public Object clone() throws CloneNotSupportedException
        {
        Bag b = (Bag)(super.clone());
        b.objs = (Object[]) objs.clone();
        return b;
        }

    /** Resizes the internal array to at least the requested
       size. */
    public void resize(int toAtLeast)
        {
        if (objs.length >= toAtLeast)  // already at least as
           big as requested
            return;

        if (objs.length * 2 > toAtLeast)  // worth doubling
            toAtLeast = objs.length * 2;

        // now resize
        Object[] newobjs = new Object[toAtLeast];
        System.arraycopy(objs,0,newobjs,0,numObjs);
        objs=newobjs;
        }

    /** Resizes the objs array to max(numObjs, desiredLength),
       unless that value is greater than or equal to objs.
       length,
        in which case no resizing is done (this operation only
           shrinks -- use resize() instead).
        This is an O(n) operation, so use it sparingly. */
    public void shrink(int desiredLength)
        {
        if (desiredLength < numObjs) desiredLength = numObjs;
        if (desiredLength >= objs.length) return;  // no reason
            to bother
        Object[] newobjs = new Object[desiredLength];
        System.arraycopy(objs,0,newobjs,0,numObjs);
        objs = newobjs;
```

```
    }

/** Returns null if the Bag is empty, else returns the
   topmost object. */
public Object top()
    {
    if (numObjs<= 0) return null;
    else return objs[numObjs-1];
    }

/** Returns null if the Bag is empty, else removes and
   returns the topmost object. */
public Object pop()
    {
    // this curious arrangement makes me small enough to be
        inlined (35 bytes; right at the limit)
    int numObjs = this.numObjs;
    if (numObjs<= 0) return null;
    Object ret = objs[--numObjs];
    objs[numObjs] = null; // let GC
    this.numObjs = numObjs;
    return ret;
    }

/** Synonym for add(obj) -- stylistically, you should add
   instead unless you
    want to think of the Bag as a stack. */
public boolean push(final Object obj)
    {
    // this curious arrangement makes me small enough to be
        inlined (35 bytes)
    int numObjs = this.numObjs;
    if (numObjs >= objs.length) doubleCapacityPlusOne();
    objs[numObjs] = obj;
    this.numObjs = numObjs+1;
    return true;
    }

public boolean add(final Object obj)
    {
    // this curious arrangement makes me small enough to be
        inlined (35 bytes)
    int numObjs = this.numObjs;
    if (numObjs >= objs.length) doubleCapacityPlusOne();
    objs[numObjs] = obj;
    this.numObjs = numObjs+1;
```

```
    return true;
    }

// private function used by add and push in order to get
   them below
// 35 bytes -- always doubles the capacity and adds one
void doubleCapacityPlusOne()
    {
    Object[] newobjs = new Object[numObjs*2+1];
    System.arraycopy(objs,0,newobjs,0,numObjs);
    objs=newobjs;
    }

public boolean contains(final Object o)
    {
    int numObjs = this.numObjs;
    Object[] objs = this.objs;
    for(int x=0;x<numObjs;x++)
        if (o==null ? objs[x]==null : o==objs[x] || o.
           equals(objs[x])) return true;
    return false;
    }

public boolean containsAll(final Collection c)
    {
    Iterator iterator = c.iterator();
    while(iterator.hasNext())
        if (!contains(iterator.next())) return false;
    return true;
    }

public Object get(final int index)
    {
    if (index>=numObjs) // || index < 0)
        throwIndexOutOfBoundsException(index);
    return objs[index];
    }

/** identical to get(index) */
public Object getValue(final int index)
    {
    if (index>=numObjs) // || index < 0)
        throwIndexOutOfBoundsException(index);
    return objs[index];
    }
```

```
public Object set(final int index, final Object element)
    {
    if (index>=numObjs) // || index < 0)
        throwIndexOutOfBoundsException(index);
    Object returnval = objs[index];
    objs[index] = element;
    return returnval;
    }

/** identical to set(index, element) */
public Object setValue(final int index, final Object
   element)
    {
    if (index>=numObjs) // || index < 0)
        throwIndexOutOfBoundsException(index);
    Object returnval = objs[index];
    objs[index] = element;
    return returnval;
    }

public boolean removeAll(final Collection c)
    {
    boolean flag = false;
    Iterator iterator = c.iterator();
    while(iterator.hasNext())
        if (remove(iterator.next())) flag = true;
    return flag;
    }

public boolean retainAll(final Collection c)
    {
    boolean flag = false;
    for(int x=0;x<numObjs;x++)
        if (!c.contains(objs[x]))
            {
            flag = true;
            remove(x);
            x--; // consider the newly-swapped-in item
            }
    return flag;
    }

/** Removes the object at the given index, shifting the
   other objects down. */
public Object removeNondestructively(final int index)
    {
```

```
    if (index>=numObjs) // || index < 0)
        throwIndexOutOfBoundsException(index);
    Object ret = objs[index];
    if (index < numObjs - 1)  // it's not the topmost
       object, must swap down
        System.arraycopy(objs, index+1, objs, index,
           numObjs - index - 1);
    objs[numObjs-1] = null;  // let GC
    numObjs--;
    return ret;
    }

/** Removes the object, moving the topmost object into its
   position. */
public boolean remove(final Object o)
    {
    int numObjs = this.numObjs;
    Object[] objs = this.objs;
    for(int x=0;x<numObjs;x++)
        if (o==null ? objs[x]==null : o==objs[x] || o.
           equals(objs[x]))
            {
            remove(x);
            return true;
            }
    return false;
    }

/** Removes multiple instantiations of an object */
public boolean removeMultiply(final Object o)
    {
    int numObjs = this.numObjs;
    Object[] objs = this.objs;
    boolean flag = false;
    for(int x=0;x<numObjs;x++)
        if (o==null ? objs[x]==null : o==objs[x] || o.
           equals(objs[x]))
            {
            flag = true;
            remove(x);
            x--;  // to check the next item swapped in...
            }
    return flag;
    }
```

```
/** Removes the object at the given index, moving the
   topmost object into its position. */
public Object remove(final int index)
    {
    int _numObjs = numObjs;
    if (index >= _numObjs) // || index < 0)
        throwIndexOutOfBoundsException(index);
    Object[] _objs = this.objs;
    Object ret = _objs[index];
    _objs[index] = _objs[_numObjs-1];
    _objs[_numObjs-1] = null;  // let GC
    numObjs--;
    return ret;
    }

protected void throwIndexOutOfBoundsException(final int
   index)
    {
    throw new IndexOutOfBoundsException(""+index);
    }

/** Removes all objects in the Bag.  This is done by
   clearing the internal array but
    not replacing it with a new, smaller one. */
public void clear()
    {
    // local variables are faster
    int len = numObjs;
    Object[] o = objs;

    for(int i = 0; i < len; i++)
        o[i] = null;  // let GC

    numObjs = 0;
    }

public Object[] toArray()
    {
    Object[] o = new Object[numObjs];
    System.arraycopy(objs,0,o,0,numObjs);
    return o;
    }

// revised for new Java protocol requirements: returned
   array must be same component
```

```
// type as the passed in array; passed in array is not used
    if it is too small;
// null pointer exception is thrown.
public Object[] toArray(Object[] o)
    {
    if (o.length < numObjs)  // will throw a null pointer
       exception (properly) if o is null
        o = (Object[])(Array.newInstance(o.getClass().
           getComponentType(), numObjs));
    else if (o.length > numObjs)
        o[numObjs] = null;
    System.arraycopy(objs,0,o,0,numObjs);
    return o;
    }


/** NOT fail-fast.  Use this method only if you're
    concerned about accessing numObjs and objs directly.
       */
public Iterator iterator()
    {
    return new BagIterator(this);
    }

/** Always returns null.  This method is to adhere to
   Indexed. */
public Class componentType()
    {
    return null;
    }

/** Sorts the bag according to the provided comparator */
public void sort(Comparator c)
    {
    Arrays.sort(objs, 0, numObjs, c);
    }

/** Replaces all elements in the bag with the provided
   object. */
public void fill(Object o)
    {
    // teeny bit faster
    Object[] objs = this.objs;
    int numObjs = this.numObjs;

    for(int x=0; x < numObjs; x++)
```

```
        objs[x] = o;
    }

/** Shuffles (randomizes the order of) the Bag */
public void shuffle(Random random)
    {
    // teeny bit faster
    Object[] objs = this.objs;
    int numObjs = this.numObjs;
    Object obj;
    int rand;

    for(int x=numObjs-1; x >= 1 ; x--)
        {
        rand = random.nextInt(x+1);
        obj = objs[x];
        objs[x] = objs[rand];
        objs[rand] = obj;
        }
    }

/** Shuffles (randomizes the order of) the Bag */
public void shuffle(util.MersenneTwisterFast random)
    {
    // teeny bit faster
    Object[] objs = this.objs;
    int numObjs = this.numObjs;
    Object obj;
    int rand;

    for(int x=numObjs-1; x >= 1 ; x--)
        {
        rand = random.nextInt(x+1);
        obj = objs[x];
        objs[x] = objs[rand];
        objs[rand] = obj;
        }
    }

/** Reverses order of the elements in the Bag */
public void reverse()
    {
    // teeny bit faster
    Object[] objs = this.objs;
    int numObjs = this.numObjs;
    int l = numObjs / 2;
```

```
        Object obj;
        for(int x=0; x < l; x++)
            {
            obj = objs[x];
            objs[x] = objs[numObjs - x - 1];
            objs[numObjs - x - 1] = obj;
            }
        }

    static class BagIterator implements Iterator, java.io.
        Serializable
        {
        int obj = 0;
        Bag bag;
        boolean canRemove = false;

        public BagIterator(Bag bag) { this.bag = bag; }

        public boolean hasNext()
            {
            return (obj < bag.numObjs);
            }
        public Object next()
            {
            if (obj >= bag.numObjs) throw new
                NoSuchElementException("No More Elements");
            canRemove = true;
            return bag.objs[obj++];
            }
        public void remove()
            {
            if (!canRemove) throw new IllegalStateException("
                remove() before next(), or remove() called twice
                ");
            // more consistent with the following line than '
                obj > bag.numObjs' would be...
            if (obj - 1 >=  bag.numObjs) throw new
                NoSuchElementException("No More Elements");
            bag.removeNondestructively(obj-1);
            obj--;
            canRemove = false;
            }
        // static inner class -- no need to add a
            serialVersionUID
        }
    }
```

Indexed.java

```
/*
 Copyright 2006 by Sean Luke and George Mason University
 Licensed under the Academic Free License version 3.0
 See the file "LICENSE" for more information
*/

package util;

/** A simple interface (simpler than List) for accessing
    random-access objects without changing their size.  Adhered
     to by Bag, IntBag, and DoubleBag */

public interface Indexed
    {
    /** Should return the base component type for this Indexed
        object, or
         null if the component type should be queried via
            getValue(index).getClass.getComponentType() */
    public Class componentType();
    public int size();
    /** Throws an IndexOutOfBoundsException if index is
        inappropriate, and IllegalArgumentException
         if the value is inappropriate.  Not called set() in
            order to be consistent with getValue(...)*/
    public Object setValue(final int index, final Object value)
        throws IndexOutOfBoundsException,
            IllegalArgumentException;
    /** Throws an IndexOutOfBoundsException if index is
        inappropriate.  Not called get() because
         this would conflict with get() methods in IntBag etc.
            which don't return objects. */
    public Object getValue(final int index)
        throws IndexOutOfBoundsException;
    }
```

MersenneTwister.java

```
package util;
import java.io.*;
import java.util.*;
```

```
/**
* <h3>MersenneTwister and MersenneTwisterFast</h3>
* <p><b>Version 16</b>, based on version MT199937(99/10/29)
* of the Mersenne Twister algorithm found at
* <a href="http://www.math.keio.ac.jp/matumoto/emt.html">
* The Mersenne Twister Home Page</a>, with the initialization
* improved using the new 2002/1/26 initialization algorithm
* By Sean Luke, October 2004.
*
* <p><b>MersenneTwister</b> is a drop-in subclass replacement
* for java.util.Random.  It is properly synchronized and
* can be used in a multithreaded environment.  On modern VMs
    such
* as HotSpot, it is approximately 1/3 slower than java.util.
    Random.
*
* <p><b>MersenneTwisterFast</b> is not a subclass of java.util
    .Random.  It has
* the same public methods as Random does, however, and it is
* algorithmically identical to MersenneTwister.
    MersenneTwisterFast
* has hard-code inlined all of its methods directly, and made
    all of them
* final (well, the ones of consequence anyway).  Further,
    these
* methods are <i>not</i> synchronized, so the same
    MersenneTwisterFast
* instance cannot be shared by multiple threads.  But all this
     helps
* MersenneTwisterFast achieve well over twice the speed of
    MersenneTwister.
* java.util.Random is about 1/3 slower than
    MersenneTwisterFast.
*
* <h3>About the Mersenne Twister</h3>
* <p>This is a Java version of the C-program for MT19937:
    Integer version.
* The MT19937 algorithm was created by Makoto Matsumoto and
    Takuji Nishimura,
* who ask: "When you use this, send an email to: matumoto@math
    .keio.ac.jp
* with an appropriate reference to your work".  Indicate that
    this
* is a translation of their algorithm into Java.
*
* <p><b>Reference. </b>
```

```
* Makato Matsumoto and Takuji Nishimura,
* "Mersenne Twister: A 623-Dimensionally Equidistributed
   Uniform
* Pseudo-Random Number Generator",
* <i>ACM Transactions on Modeling and. Computer Simulation,</i
   >
* Vol. 8, No. 1, January 1998, pp 3--30.
*
* <h3>About this Version</h3>
*
* <p><b>Changes Since V15:</b> Added serialVersionUID to quiet
    compiler warnings
* from Sun's overly verbose compilers as of JDK 1.5.
*
* <p><b>Changes Since V14:</b> made strictfp, with StrictMath.
   log and StrictMath.sqrt
* in nextGaussian instead of Math.log and Math.sqrt. This is
   largely just to be safe,
* as it presently makes no difference in the speed,
   correctness, or results of the
* algorithm.
*
* <p><b>Changes Since V13:</b> clone() method
   CloneNotSupportedException removed.
*
* <p><b>Changes Since V12:</b> clone() method added.
*
* <p><b>Changes Since V11:</b> stateEquals(...) method added.
    MersenneTwisterFast
* is equal to other MersenneTwisterFasts with identical state;
    likewise
* MersenneTwister is equal to other MersenneTwister with
   identical state.
* This isn't equals(...) because that requires a contract of
   immutability
* to compare by value.
*
* <p><b>Changes Since V10:</b> A documentation error suggested
    that
* setSeed(int[]) required an int[] array 624 long. In fact,
   the array
* can be any non-zero length. The new version also checks for
    this fact.
*
* <p><b>Changes Since V9:</b> readState(stream) and writeState
   (stream)
```

```
* provided.
*
* <p><b>Changes Since V8:</b> setSeed(int) was only using the
    first 28 bits
* of the seed; it should have been 32 bits.  For small-number
    seeds the
* behavior is identical.
*
* <p><b>Changes Since V7:</b> A documentation error in
    MersenneTwisterFast
* (but not MersenneTwister) stated that nextDouble selects
    uniformly from
* the full-open interval [0,1].  It does not.  nextDouble's
    contract is
* identical across MersenneTwisterFast, MersenneTwister, and
    java.util.Random,
* namely, selection in the half-open interval [0,1).  That is,
    1.0 should
* not be returned.  A similar contract exists in nextFloat.
*
* <p><b>Changes Since V6:</b> License has changed from LGPL to
    BSD.
* New timing information to compare against
* java.util.Random.  Recent versions of HotSpot have helped
    Random increase
* in speed to the point where it is faster than
    MersenneTwister but slower
* than MersenneTwisterFast (which should be the case, as it's
    a less complex
* algorithm but is synchronized).
*
* <p><b>Changes Since V5:</b> New empty constructor made to
    work the same
* as java.util.Random -- namely, it seeds based on the current
    time in
* milliseconds.
*
* <p><b>Changes Since V4:</b> New initialization algorithms.
    See
* (see <a href="http://www.math.keio.ac.jp/matumoto/MT2002/
    emt19937ar.html"</a>
* http://www.math.keio.ac.jp/matumoto/MT2002/emt19937ar.html</
    a>)
*
* <p>The MersenneTwister code is based on standard MT19937 C/C
    ++
```

```
* code by Takuji Nishimura,
* with suggestions from Topher Cooper and Marc Rieffel, July
    1997.
* The code was originally translated into Java by Michael
    Lecuyer,
* January 1999, and the original code is Copyright (c) 1999 by
     Michael Lecuyer.
*
* <h3>Java notes</h3>
*
* <p>This implementation implements the bug fixes made
* in Java 1.2's version of Random, which means it can be used
    with
* earlier versions of Java.  See
* <a href="http://www.javasoft.com/products/jdk/1.2/docs/api/
    java/util/Random.html">
* the JDK 1.2 java.util.Random documentation</a> for further
    documentation
* on the random-number generation contracts made.
    Additionally, there's
* an undocumented bug in the JDK java.util.Random.nextBytes()
    method,
* which this code fixes.
*
* <p> Just like java.util.Random, this
* generator accepts a long seed but doesn't use all of it.
    java.util.Random
* uses 48 bits.  The Mersenne Twister instead uses 32 bits (
    int size).
* So it's best if your seed does not exceed the int range.
*
* <p>MersenneTwister can be used reliably
* on JDK version 1.1.5 or above.  Earlier Java versions have
    serious bugs in
* java.util.Random; only MersenneTwisterFast (and not
    MersenneTwister nor
* java.util.Random) should be used with them.
*
* <h3>License</h3>
*
* Copyright (c) 2003 by Sean Luke. <br>
* Portions copyright (c) 1993 by Michael Lecuyer. <br>
* All rights reserved. <br>
*
* <p>Redistribution and use in source and binary forms, with
    or without
```

```
* modification, are permitted provided that the following
    conditions are met:
* <ul>
* <li> Redistributions of source code must retain the above
    copyright notice,
* this list of conditions and the following disclaimer.
* <li> Redistributions in binary form must reproduce the above
     copyright notice,
* this list of conditions and the following disclaimer in the
    documentation
* and/or other materials provided with the distribution.
* <li> Neither the name of the copyright owners, their
    employers, nor the
* names of its contributors may be used to endorse or promote
    products
* derived from this software without specific prior written
    permission.
* </ul>
* <p>THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND
    CONTRIBUTORS "AS IS"
* AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
    LIMITED TO, THE
* IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A
    PARTICULAR PURPOSE ARE
* DISCLAIMED.  IN NO EVENT SHALL THE COPYRIGHT OWNERS OR
    CONTRIBUTORS BE
* LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
    EXEMPLARY, OR
* CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
    PROCUREMENT OF
* SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS;
     OR BUSINESS
* INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY,
     WHETHER IN
* CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR
     OTHERWISE)
* ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF
    ADVISED OF THE
* POSSIBILITY OF SUCH DAMAGE.
*
@version 16
*/

// Note: this class is hard-inlined in all of its methods.
    This makes some of
```

```
// the methods well-nigh unreadable in their complexity.  In
   fact, the Mersenne
// Twister is fairly easy code to understand: if you're trying
    to get a handle
// on the code, I strongly suggest looking at MersenneTwister.
   java first.
// -- Sean

public strictfp class MersenneTwisterFast implements
   Serializable, Cloneable
   {
   // Serialization
   private static final long serialVersionUID =
      -8219700664442619525L;  // locked as of Version 15

   // Period parameters
   private static final int N = 624;
   private static final int M = 397;
   private static final int MATRIX_A = 0x9908b0df;   //
      private static final * constant vector a
   private static final int UPPER_MASK = 0x80000000; // most
      significant w-r bits
   private static final int LOWER_MASK = 0x7fffffff; // least
      significant r bits


   // Tempering parameters
   private static final int TEMPERING_MASK_B = 0x9d2c5680;
   private static final int TEMPERING_MASK_C = 0xefc60000;

   private int mt[]; // the array for the state vector
   private int mti; // mti==N+1 means mt[N] is not initialized
   private int mag01[];

   // a good initial seed (of int size, though stored in a
      long)
   //private static final long GOOD_SEED = 4357;

   private double __nextNextGaussian;
   private boolean __haveNextNextGaussian;

   /* We're overriding all internal data, to my knowledge, so
      this should be okay */
   public Object clone()
       {
       try
```

```
            {
            MersenneTwisterFast f = (MersenneTwisterFast)(super
                .clone());
            f.mt = (int[])(mt.clone());
            f.mag01 = (int[])(mag01.clone());
            return f;
            }
        catch (CloneNotSupportedException e) { throw new
            InternalError(); } // should never happen
        }

    public boolean stateEquals(Object o)
        {
        if (o==this) return true;
        if (o == null || !(o instanceof MersenneTwisterFast))
            return false;
        MersenneTwisterFast other = (MersenneTwisterFast) o;
        if (mti != other.mti) return false;
        for(int x=0;x<mag01.length;x++)
            if (mag01[x] != other.mag01[x]) return false;
        for(int x=0;x<mt.length;x++)
            if (mt[x] != other.mt[x]) return false;
        return true;
        }

    /** Reads the entire state of the MersenneTwister RNG from
        the stream */
    public void readState(DataInputStream stream) throws
        IOException
        {
        int len = mt.length;
        for(int x=0;x<len;x++) mt[x] = stream.readInt();

        len = mag01.length;
        for(int x=0;x<len;x++) mag01[x] = stream.readInt();

        mti = stream.readInt();
        __nextNextGaussian = stream.readDouble();
        __haveNextNextGaussian = stream.readBoolean();
        }

    /** Writes the entire state of the MersenneTwister RNG to
        the stream */
    public void writeState(DataOutputStream stream) throws
        IOException
        {
```

```
        int len = mt.length;
        for(int x=0;x<len;x++) stream.writeInt(mt[x]);

        len = mag01.length;
        for(int x=0;x<len;x++) stream.writeInt(mag01[x]);

        stream.writeInt(mti);
        stream.writeDouble(__nextNextGaussian);
        stream.writeBoolean(__haveNextNextGaussian);
        }

    /**
     * Constructor using the default seed.
     */
    public MersenneTwisterFast()
        {
        this(System.currentTimeMillis());
        }

    /**
     * Constructor using a given seed.  Though you pass this
        seed in
     * as a long, it's best to make sure it's actually an
        integer.
     *
     */
    public MersenneTwisterFast(final long seed)
        {
        setSeed(seed);
        }


    /**
     * Constructor using an array of integers as seed.
     * Your array must have a non-zero length.  Only the first
        624 integers
     * in the array are used; if the array is shorter than this
         then
     * integers are repeatedly used in a wrap-around fashion.
     */
    public MersenneTwisterFast(final int[] array)
        {
        setSeed(array);
        }
```

```
/**
 * Initalize the pseudo random number generator.  Don't
 * pass in a long that's bigger than an int (Mersenne
    Twister
 * only uses the first 32 bits for its seed).
 */

synchronized public void setSeed(final long seed)
    {
    // Due to a bug in java.util.Random clear up to 1.2, we
       're
    // doing our own Gaussian variable.
    __haveNextNextGaussian = false;

    mt = new int[N];

    mag01 = new int[2];
    mag01[0] = 0x0;
    mag01[1] = MATRIX_A;

    mt[0]= (int)(seed & 0xffffffff);
    for (mti=1; mti<N; mti++)
        {
        mt[mti] =
            (1812433253 * (mt[mti-1] ^ (mt[mti-1] >>> 30))
               + mti);
        /* See Knuth TAOCP Vol2. 3rd Ed. P.106 for
           multiplier. */
        /* In the previous versions, MSBs of the seed
           affect   */
        /* only MSBs of the array mt[].
                                    */
        /* 2002/01/09 modified by Makoto Matsumoto
                     */
        mt[mti] &= 0xffffffff;
        /* for >32 bit machines */
        }
    }


/**
 * Sets the seed of the MersenneTwister using an array of
    integers.
 * Your array must have a non-zero length.  Only the first
    624 integers
```

```
     * in the array are used; if the array is shorter than this
         then
     * integers are repeatedly used in a wrap-around fashion.
     */

    synchronized public void setSeed(final int[] array)
        {
        if (array.length == 0)
            throw new IllegalArgumentException("Array length
                must be greater than zero");
        int i, j, k;
        setSeed(19650218);
        i=1; j=0;
        k = (N>array.length ? N : array.length);
        for (; k!=0; k--)
            {
            mt[i] = (mt[i] ^ ((mt[i-1] ^ (mt[i-1] >>> 30)) *
                1664525)) + array[j] + j; /* non linear */
            mt[i] &= 0xffffffff; /* for WORDSIZE > 32 machines
                */
            i++;
            j++;
            if (i>=N) { mt[0] = mt[N-1]; i=1; }
            if (j>=array.length) j=0;
            }
        for (k=N-1; k!=0; k--)
            {
            mt[i] = (mt[i] ^ ((mt[i-1] ^ (mt[i-1] >>> 30)) *
                1566083941)) - i; /* non linear */
            mt[i] &= 0xffffffff; /* for WORDSIZE > 32 machines
                */
            i++;
            if (i>=N)
                {
                mt[0] = mt[N-1]; i=1;
                }
            }
        mt[0] = 0x80000000; /* MSB is 1; assuring non-zero
            initial array */
        }


    public final int nextInt()
        {
        int y;
```

```
        if (mti >= N)    // generate N words at one time
            {
            int kk;
            final int[] mt = this.mt; // locals are slightly
               faster
            final int[] mag01 = this.mag01; // locals are
               slightly faster

            for (kk = 0; kk < N - M; kk++)
                {
                y = (mt[kk] & UPPER_MASK) | (mt[kk+1] &
                   LOWER_MASK);
                mt[kk] = mt[kk+M] ^ (y >>> 1) ^ mag01[y & 0x1];
                }
            for (; kk < N-1; kk++)
                {
                y = (mt[kk] & UPPER_MASK) | (mt[kk+1] &
                   LOWER_MASK);
                mt[kk] = mt[kk+(M-N)] ^ (y >>> 1) ^ mag01[y & 0
                   x1];
                }
            y = (mt[N-1] & UPPER_MASK) | (mt[0] & LOWER_MASK);
            mt[N-1] = mt[M-1] ^ (y >>> 1) ^ mag01[y & 0x1];

            mti = 0;
            }

        y = mt[mti++];
        y ^= y >>> 11;                                  //
           TEMPERING_SHIFT_U(y)
        y ^= (y << 7) & TEMPERING_MASK_B;               //
           TEMPERING_SHIFT_S(y)
        y ^= (y << 15) & TEMPERING_MASK_C;              //
           TEMPERING_SHIFT_T(y)
        y ^= (y >>> 18);                                //
           TEMPERING_SHIFT_L(y)

        return y;
        }



    public final short nextShort()
        {
        int y;
```

```
        if (mti >= N)    // generate N words at one time
            {
            int kk;
            final int[] mt = this.mt; // locals are slightly
               faster
            final int[] mag01 = this.mag01; // locals are
               slightly faster

            for (kk = 0; kk < N - M; kk++)
                {
                y = (mt[kk] & UPPER_MASK) | (mt[kk+1] &
                   LOWER_MASK);
                mt[kk] = mt[kk+M] ^ (y >>> 1) ^ mag01[y & 0x1];
                }
            for (; kk < N-1; kk++)
                {
                y = (mt[kk] & UPPER_MASK) | (mt[kk+1] &
                   LOWER_MASK);
                mt[kk] = mt[kk+(M-N)] ^ (y >>> 1) ^ mag01[y & 0
                   x1];
                }
            y = (mt[N-1] & UPPER_MASK) | (mt[0] & LOWER_MASK);
            mt[N-1] = mt[M-1] ^ (y >>> 1) ^ mag01[y & 0x1];

            mti = 0;
            }

        y = mt[mti++];
        y ^= y >>> 11;                                  //
           TEMPERING_SHIFT_U(y)
        y ^= (y << 7) & TEMPERING_MASK_B;               //
           TEMPERING_SHIFT_S(y)
        y ^= (y << 15) & TEMPERING_MASK_C;              //
           TEMPERING_SHIFT_T(y)
        y ^= (y >>> 18);                                //
           TEMPERING_SHIFT_L(y)

        return (short)(y >>> 16);
        }



    public final char nextChar()
        {
        int y;
```

```
        if (mti >= N)    // generate N words at one time
            {
            int kk;
            final int[] mt = this.mt; // locals are slightly
               faster
            final int[] mag01 = this.mag01; // locals are
               slightly faster

            for (kk = 0; kk < N - M; kk++)
                {
                y = (mt[kk] & UPPER_MASK) | (mt[kk+1] &
                   LOWER_MASK);
                mt[kk] = mt[kk+M] ^ (y >>> 1) ^ mag01[y & 0x1];
                }
            for (; kk < N-1; kk++)
                {
                y = (mt[kk] & UPPER_MASK) | (mt[kk+1] &
                   LOWER_MASK);
                mt[kk] = mt[kk+(M-N)] ^ (y >>> 1) ^ mag01[y & 0
                   x1];
                }
            y = (mt[N-1] & UPPER_MASK) | (mt[0] & LOWER_MASK);
            mt[N-1] = mt[M-1] ^ (y >>> 1) ^ mag01[y & 0x1];

            mti = 0;
            }

        y = mt[mti++];
        y ^= y >>> 11;                                  //
           TEMPERING_SHIFT_U(y)
        y ^= (y << 7) & TEMPERING_MASK_B;               //
           TEMPERING_SHIFT_S(y)
        y ^= (y << 15) & TEMPERING_MASK_C;              //
           TEMPERING_SHIFT_T(y)
        y ^= (y >>> 18);                                //
           TEMPERING_SHIFT_L(y)

        return (char)(y >>> 16);
        }


    public final boolean nextBoolean()
        {
        int y;

        if (mti >= N)    // generate N words at one time
```

```
        {
        int kk;
        final int[] mt = this.mt; // locals are slightly
            faster
        final int[] mag01 = this.mag01; // locals are
            slightly faster

        for (kk = 0; kk < N - M; kk++)
            {
            y = (mt[kk] & UPPER_MASK) | (mt[kk+1] &
                LOWER_MASK);
            mt[kk] = mt[kk+M] ^ (y >>> 1) ^ mag01[y & 0x1];
            }
        for (; kk < N-1; kk++)
            {
            y = (mt[kk] & UPPER_MASK) | (mt[kk+1] &
                LOWER_MASK);
            mt[kk] = mt[kk+(M-N)] ^ (y >>> 1) ^ mag01[y & 0
                x1];
            }
        y = (mt[N-1] & UPPER_MASK) | (mt[0] & LOWER_MASK);
        mt[N-1] = mt[M-1] ^ (y >>> 1) ^ mag01[y & 0x1];

        mti = 0;
        }

    y = mt[mti++];
    y ^= y >>> 11;                          //
        TEMPERING_SHIFT_U(y)
    y ^= (y << 7) & TEMPERING_MASK_B;       //
        TEMPERING_SHIFT_S(y)
    y ^= (y << 15) & TEMPERING_MASK_C;      //
        TEMPERING_SHIFT_T(y)
    y ^= (y >>> 18);                        //
        TEMPERING_SHIFT_L(y)

    return (boolean)((y >>> 31) != 0);
    }



/** This generates a coin flip with a probability <tt>
    probability</tt>
    of returning true, else returning false. <tt>
        probability</tt> must
```

```
    be between 0.0 and 1.0, inclusive.  Not as precise a
       random real
    event as nextBoolean(double), but twice as fast. To
       explicitly
    use this, remember you may need to cast to float first.
        */

  public final boolean nextBoolean(final float probability)
    {
    int y;

    if (probability < 0.0f || probability > 1.0f)
        throw new IllegalArgumentException ("probability␣
           must␣be␣between␣0.0␣and␣1.0␣inclusive.");
    if (probability==0.0f) return false;             // fix
       half-open issues
    else if (probability==1.0f) return true;         // fix
       half-open issues
    if (mti >= N)   // generate N words at one time
        {
        int kk;
        final int[] mt = this.mt; // locals are slightly
           faster
        final int[] mag01 = this.mag01; // locals are
           slightly faster

        for (kk = 0; kk < N - M; kk++)
            {
            y = (mt[kk] & UPPER_MASK) | (mt[kk+1] &
               LOWER_MASK);
            mt[kk] = mt[kk+M] ^ (y >>> 1) ^ mag01[y & 0x1];
            }
        for (; kk < N-1; kk++)
            {
            y = (mt[kk] & UPPER_MASK) | (mt[kk+1] &
               LOWER_MASK);
            mt[kk] = mt[kk+(M-N)] ^ (y >>> 1) ^ mag01[y & 0
               x1];
            }
        y = (mt[N-1] & UPPER_MASK) | (mt[0] & LOWER_MASK);
        mt[N-1] = mt[M-1] ^ (y >>> 1) ^ mag01[y & 0x1];

        mti = 0;
        }

    y = mt[mti++];
```

```
        y ^= y >>> 11;                          //
            TEMPERING_SHIFT_U(y)
        y ^= (y << 7) & TEMPERING_MASK_B;       //
            TEMPERING_SHIFT_S(y)
        y ^= (y << 15) & TEMPERING_MASK_C;      //
            TEMPERING_SHIFT_T(y)
        y ^= (y >>> 18);                        //
            TEMPERING_SHIFT_L(y)

        return (y >>> 8) / ((float)(1 << 24)) < probability;
        }


    /** This generates a coin flip with a probability <tt>
        probability</tt>
        of returning true, else returning false. <tt>
            probability</tt> must
        be between 0.0 and 1.0, inclusive. */

    public final boolean nextBoolean(final double probability)
        {
        int y;
        int z;

        if (probability < 0.0 || probability > 1.0)
            throw new IllegalArgumentException ("probability␣
                must␣be␣between␣0.0␣and␣1.0␣inclusive.");
        if (probability==0.0) return false;             // fix
            half-open issues
        else if (probability==1.0) return true; // fix half-
            open issues
        if (mti >= N)   // generate N words at one time
            {
            int kk;
            final int[] mt = this.mt; // locals are slightly
                faster
            final int[] mag01 = this.mag01; // locals are
                slightly faster

            for (kk = 0; kk < N - M; kk++)
                {
                y = (mt[kk] & UPPER_MASK) | (mt[kk+1] &
                    LOWER_MASK);
                mt[kk] = mt[kk+M] ^ (y >>> 1) ^ mag01[y & 0x1];
                }
            for (; kk < N-1; kk++)
```

```
            {
            y = (mt[kk] & UPPER_MASK) | (mt[kk+1] &
                LOWER_MASK);
            mt[kk] = mt[kk+(M-N)] ^ (y >>> 1) ^ mag01[y & 0
                x1];
            }
        y = (mt[N-1] & UPPER_MASK) | (mt[0] & LOWER_MASK);
        mt[N-1] = mt[M-1] ^ (y >>> 1) ^ mag01[y & 0x1];

        mti = 0;
        }

    y = mt[mti++];
    y ^= y >>> 11;                              //
        TEMPERING_SHIFT_U(y)
    y ^= (y << 7) & TEMPERING_MASK_B;           //
        TEMPERING_SHIFT_S(y)
    y ^= (y << 15) & TEMPERING_MASK_C;          //
        TEMPERING_SHIFT_T(y)
    y ^= (y >>> 18);                            //
        TEMPERING_SHIFT_L(y)

    if (mti >= N)    // generate N words at one time
        {
        int kk;
        final int[] mt = this.mt; // locals are slightly
            faster
        final int[] mag01 = this.mag01; // locals are
            slightly faster

        for (kk = 0; kk < N - M; kk++)
            {
            z = (mt[kk] & UPPER_MASK) | (mt[kk+1] &
                LOWER_MASK);
            mt[kk] = mt[kk+M] ^ (z >>> 1) ^ mag01[z & 0x1];
            }
        for (; kk < N-1; kk++)
            {
            z = (mt[kk] & UPPER_MASK) | (mt[kk+1] &
                LOWER_MASK);
            mt[kk] = mt[kk+(M-N)] ^ (z >>> 1) ^ mag01[z & 0
                x1];
            }
        z = (mt[N-1] & UPPER_MASK) | (mt[0] & LOWER_MASK);
        mt[N-1] = mt[M-1] ^ (z >>> 1) ^ mag01[z & 0x1];
```

```
            mti = 0;
            }

        z = mt[mti++];
        z ^= z >>> 11;                              //
            TEMPERING_SHIFT_U(z)
        z ^= (z << 7) & TEMPERING_MASK_B;           //
            TEMPERING_SHIFT_S(z)
        z ^= (z << 15) & TEMPERING_MASK_C;          //
            TEMPERING_SHIFT_T(z)
        z ^= (z >>> 18);                            //
            TEMPERING_SHIFT_L(z)

        /* derived from nextDouble documentation in jdk 1.2
            docs, see top */
        return ((((long)(y >>> 6)) << 27) + (z >>> 5)) / (
            double)(1L << 53) < probability;
        }


    public final byte nextByte()
        {
        int y;

        if (mti >= N)   // generate N words at one time
            {
            int kk;
            final int[] mt = this.mt; // locals are slightly
                faster
            final int[] mag01 = this.mag01; // locals are
                slightly faster

            for (kk = 0; kk < N - M; kk++)
                {
                y = (mt[kk] & UPPER_MASK) | (mt[kk+1] &
                    LOWER_MASK);
                mt[kk] = mt[kk+M] ^ (y >>> 1) ^ mag01[y & 0x1];
                }
            for (; kk < N-1; kk++)
                {
                y = (mt[kk] & UPPER_MASK) | (mt[kk+1] &
                    LOWER_MASK);
                mt[kk] = mt[kk+(M-N)] ^ (y >>> 1) ^ mag01[y & 0
                    x1];
                }
            y = (mt[N-1] & UPPER_MASK) | (mt[0] & LOWER_MASK);
```

```
            mt[N-1] = mt[M-1] ^ (y >>> 1) ^ mag01[y & 0x1];

            mti = 0;
            }

        y = mt[mti++];
        y ^= y >>> 11;                          //
            TEMPERING_SHIFT_U(y)
        y ^= (y << 7) & TEMPERING_MASK_B;       //
            TEMPERING_SHIFT_S(y)
        y ^= (y << 15) & TEMPERING_MASK_C;      //
            TEMPERING_SHIFT_T(y)
        y ^= (y >>> 18);                        //
            TEMPERING_SHIFT_L(y)

        return (byte)(y >>> 24);
        }


    public final void nextBytes(byte[] bytes)
        {
        int y;

        for (int x=0;x<bytes.length;x++)
            {
            if (mti >= N)   // generate N words at one time
                {
                int kk;
                final int[] mt = this.mt; // locals are
                    slightly faster
                final int[] mag01 = this.mag01; // locals are
                    slightly faster

                for (kk = 0; kk < N - M; kk++)
                    {
                    y = (mt[kk] & UPPER_MASK) | (mt[kk+1] &
                        LOWER_MASK);
                    mt[kk] = mt[kk+M] ^ (y >>> 1) ^ mag01[y & 0
                        x1];
                    }
                for (; kk < N-1; kk++)
                    {
                    y = (mt[kk] & UPPER_MASK) | (mt[kk+1] &
                        LOWER_MASK);
                    mt[kk] = mt[kk+(M-N)] ^ (y >>> 1) ^ mag01[y
                         & 0x1];
```

```
                }
            y = (mt[N-1] & UPPER_MASK) | (mt[0] &
                LOWER_MASK);
            mt[N-1] = mt[M-1] ^ (y >>> 1) ^ mag01[y & 0x1];

            mti = 0;
            }

        y = mt[mti++];
        y ^= y >>> 11;                          //
            TEMPERING_SHIFT_U(y)
        y ^= (y << 7) & TEMPERING_MASK_B;       //
            TEMPERING_SHIFT_S(y)
        y ^= (y << 15) & TEMPERING_MASK_C;      //
            TEMPERING_SHIFT_T(y)
        y ^= (y >>> 18);                        //
            TEMPERING_SHIFT_L(y)

        bytes[x] = (byte)(y >>> 24);
        }
    }


public final long nextLong()
    {
    int y;
    int z;

    if (mti >= N)   // generate N words at one time
        {
        int kk;
        final int[] mt = this.mt; // locals are slightly
            faster
        final int[] mag01 = this.mag01; // locals are
            slightly faster

        for (kk = 0; kk < N - M; kk++)
            {
            y = (mt[kk] & UPPER_MASK) | (mt[kk+1] &
                LOWER_MASK);
            mt[kk] = mt[kk+M] ^ (y >>> 1) ^ mag01[y & 0x1];
            }
        for (; kk < N-1; kk++)
            {
            y = (mt[kk] & UPPER_MASK) | (mt[kk+1] &
                LOWER_MASK);
```

```
            mt[kk] = mt[kk+(M-N)] ^ (y >>> 1) ^ mag01[y & 0
                x1];
            }
        y = (mt[N-1] & UPPER_MASK) | (mt[0] & LOWER_MASK);
        mt[N-1] = mt[M-1] ^ (y >>> 1) ^ mag01[y & 0x1];

        mti = 0;
        }

    y = mt[mti++];
    y ^= y >>> 11;                              //
        TEMPERING_SHIFT_U(y)
    y ^= (y << 7) & TEMPERING_MASK_B;           //
        TEMPERING_SHIFT_S(y)
    y ^= (y << 15) & TEMPERING_MASK_C;          //
        TEMPERING_SHIFT_T(y)
    y ^= (y >>> 18);                            //
        TEMPERING_SHIFT_L(y)

    if (mti >= N)    // generate N words at one time
        {
        int kk;
        final int[] mt = this.mt; // locals are slightly
            faster
        final int[] mag01 = this.mag01; // locals are
            slightly faster

        for (kk = 0; kk < N - M; kk++)
            {
            z = (mt[kk] & UPPER_MASK) | (mt[kk+1] &
                LOWER_MASK);
            mt[kk] = mt[kk+M] ^ (z >>> 1) ^ mag01[z & 0x1];
            }
        for (; kk < N-1; kk++)
            {
            z = (mt[kk] & UPPER_MASK) | (mt[kk+1] &
                LOWER_MASK);
            mt[kk] = mt[kk+(M-N)] ^ (z >>> 1) ^ mag01[z & 0
                x1];
            }
        z = (mt[N-1] & UPPER_MASK) | (mt[0] & LOWER_MASK);
        mt[N-1] = mt[M-1] ^ (z >>> 1) ^ mag01[z & 0x1];

        mti = 0;
        }
```

```
        z = mt[mti++];
        z ^= z >>> 11;                              //
            TEMPERING_SHIFT_U(z)
        z ^= (z << 7) & TEMPERING_MASK_B;           //
            TEMPERING_SHIFT_S(z)
        z ^= (z << 15) & TEMPERING_MASK_C;          //
            TEMPERING_SHIFT_T(z)
        z ^= (z >>> 18);                            //
            TEMPERING_SHIFT_L(z)

        return (((long)y) << 32) + (long)z;
        }



    /** Returns a long drawn uniformly from 0 to n-1.  Suffice
        it to say,
        n must be > 0, or an IllegalArgumentException is raised
            . */
    public final long nextLong(final long n)
        {
        if (n<=0)
            throw new IllegalArgumentException("n must be
                positive, got: " + n);

        long bits, val;
        do
            {
            int y;
            int z;

            if (mti >= N)   // generate N words at one time
                {
                int kk;
                final int[] mt = this.mt; // locals are
                    slightly faster
                final int[] mag01 = this.mag01; // locals are
                    slightly faster

                for (kk = 0; kk < N - M; kk++)
                    {
                    y = (mt[kk] & UPPER_MASK) | (mt[kk+1] &
                        LOWER_MASK);
                    mt[kk] = mt[kk+M] ^ (y >>> 1) ^ mag01[y & 0
                        x1];
                    }
```

```
        for (; kk < N-1; kk++)
            {
            y = (mt[kk] & UPPER_MASK) | (mt[kk+1] &
                LOWER_MASK);
            mt[kk] = mt[kk+(M-N)] ^ (y >>> 1) ^ mag01[y
                & 0x1];
            }
        y = (mt[N-1] & UPPER_MASK) | (mt[0] &
            LOWER_MASK);
        mt[N-1] = mt[M-1] ^ (y >>> 1) ^ mag01[y & 0x1];

        mti = 0;
        }

    y = mt[mti++];
    y ^= y >>> 11;                              //
        TEMPERING_SHIFT_U(y)
    y ^= (y << 7) & TEMPERING_MASK_B;           //
        TEMPERING_SHIFT_S(y)
    y ^= (y << 15) & TEMPERING_MASK_C;          //
        TEMPERING_SHIFT_T(y)
    y ^= (y >>> 18);                            //
        TEMPERING_SHIFT_L(y)

    if (mti >= N)    // generate N words at one time
        {
        int kk;
        final int[] mt = this.mt; // locals are
            slightly faster
        final int[] mag01 = this.mag01; // locals are
            slightly faster

        for (kk = 0; kk < N - M; kk++)
            {
            z = (mt[kk] & UPPER_MASK) | (mt[kk+1] &
                LOWER_MASK);
            mt[kk] = mt[kk+M] ^ (z >>> 1) ^ mag01[z & 0
                x1];
            }
        for (; kk < N-1; kk++)
            {
            z = (mt[kk] & UPPER_MASK) | (mt[kk+1] &
                LOWER_MASK);
            mt[kk] = mt[kk+(M-N)] ^ (z >>> 1) ^ mag01[z
                & 0x1];
            }
```

```
                z = (mt[N-1] & UPPER_MASK) | (mt[0] &
                    LOWER_MASK);
                mt[N-1] = mt[M-1] ^ (z >>> 1) ^ mag01[z & 0x1];

                mti = 0;
                }

            z = mt[mti++];
            z ^= z >>> 11;                              //
                TEMPERING_SHIFT_U(z)
            z ^= (z << 7) & TEMPERING_MASK_B;           //
                TEMPERING_SHIFT_S(z)
            z ^= (z << 15) & TEMPERING_MASK_C;          //
                TEMPERING_SHIFT_T(z)
            z ^= (z >>> 18);                            //
                TEMPERING_SHIFT_L(z)

            bits = (((((long)y) << 32) + (long)z) >>> 1);
            val = bits % n;
            } while (bits - val + (n-1) < 0);
        return val;
        }

    /** Returns a random double in the half-open range from
        [0.0,1.0).  Thus 0.0 is a valid
        result but 1.0 is not. */
    public final double nextDouble()
        {
        int y;
        int z;

        if (mti >= N)   // generate N words at one time
            {
            int kk;
            final int[] mt = this.mt; // locals are slightly
                faster
            final int[] mag01 = this.mag01; // locals are
                slightly faster

            for (kk = 0; kk < N - M; kk++)
                {
                y = (mt[kk] & UPPER_MASK) | (mt[kk+1] &
                    LOWER_MASK);
                mt[kk] = mt[kk+M] ^ (y >>> 1) ^ mag01[y & 0x1];
                }
            for (; kk < N-1; kk++)
```

```
            {
            y = (mt[kk] & UPPER_MASK) | (mt[kk+1] &
                LOWER_MASK);
            mt[kk] = mt[kk+(M-N)] ^ (y >>> 1) ^ mag01[y & 0
                x1];
            }
        y = (mt[N-1] & UPPER_MASK) | (mt[0] & LOWER_MASK);
        mt[N-1] = mt[M-1] ^ (y >>> 1) ^ mag01[y & 0x1];

        mti = 0;
        }

    y = mt[mti++];
    y ^= y >>> 11;                              //
        TEMPERING_SHIFT_U(y)
    y ^= (y << 7) & TEMPERING_MASK_B;           //
        TEMPERING_SHIFT_S(y)
    y ^= (y << 15) & TEMPERING_MASK_C;          //
        TEMPERING_SHIFT_T(y)
    y ^= (y >>> 18);                            //
        TEMPERING_SHIFT_L(y)

    if (mti >= N)    // generate N words at one time
        {
        int kk;
        final int[] mt = this.mt; // locals are slightly
            faster
        final int[] mag01 = this.mag01; // locals are
            slightly faster

        for (kk = 0; kk < N - M; kk++)
            {
            z = (mt[kk] & UPPER_MASK) | (mt[kk+1] &
                LOWER_MASK);
            mt[kk] = mt[kk+M] ^ (z >>> 1) ^ mag01[z & 0x1];
            }
        for (; kk < N-1; kk++)
            {
            z = (mt[kk] & UPPER_MASK) | (mt[kk+1] &
                LOWER_MASK);
            mt[kk] = mt[kk+(M-N)] ^ (z >>> 1) ^ mag01[z & 0
                x1];
            }
        z = (mt[N-1] & UPPER_MASK) | (mt[0] & LOWER_MASK);
        mt[N-1] = mt[M-1] ^ (z >>> 1) ^ mag01[z & 0x1];
```

```
            mti = 0;
            }

        z = mt[mti++];
        z ^= z >>> 11;                              //
            TEMPERING_SHIFT_U(z)
        z ^= (z << 7) & TEMPERING_MASK_B;           //
            TEMPERING_SHIFT_S(z)
        z ^= (z << 15) & TEMPERING_MASK_C;          //
            TEMPERING_SHIFT_T(z)
        z ^= (z >>> 18);                            //
            TEMPERING_SHIFT_L(z)

        /* derived from nextDouble documentation in jdk 1.2
            docs, see top */
        return ((((long)(y >>> 6)) << 27) + (z >>> 5)) / (
            double)(1L << 53);
        }




public final double nextGaussian()
    {
    if (__haveNextNextGaussian)
        {
        __haveNextNextGaussian = false;
        return __nextNextGaussian;
        }
    else
        {
        double v1, v2, s;
        do
            {
            int y;
            int z;
            int a;
            int b;

            if (mti >= N)    // generate N words at one time
                {
                int kk;
                final int[] mt = this.mt; // locals are
                    slightly faster
```

```
        final int[] mag01 = this.mag01; // locals
            are slightly faster

        for (kk = 0; kk < N - M; kk++)
            {
            y = (mt[kk] & UPPER_MASK) | (mt[kk+1] &
                LOWER_MASK);
            mt[kk] = mt[kk+M] ^ (y >>> 1) ^ mag01[y
                & 0x1];
            }
        for (; kk < N-1; kk++)
            {
            y = (mt[kk] & UPPER_MASK) | (mt[kk+1] &
                LOWER_MASK);
            mt[kk] = mt[kk+(M-N)] ^ (y >>> 1) ^
                mag01[y & 0x1];
            }
        y = (mt[N-1] & UPPER_MASK) | (mt[0] &
            LOWER_MASK);
        mt[N-1] = mt[M-1] ^ (y >>> 1) ^ mag01[y & 0
            x1];

        mti = 0;
        }

    y = mt[mti++];
    y ^= y >>> 11;                              //
        TEMPERING_SHIFT_U(y)
    y ^= (y << 7) & TEMPERING_MASK_B;           //
        TEMPERING_SHIFT_S(y)
    y ^= (y << 15) & TEMPERING_MASK_C;          //
        TEMPERING_SHIFT_T(y)
    y ^= (y >>> 18);                            //
        TEMPERING_SHIFT_L(y)

    if (mti >= N)    // generate N words at one time
        {
        int kk;
        final int[] mt = this.mt; // locals are
            slightly faster
        final int[] mag01 = this.mag01; // locals
            are slightly faster

        for (kk = 0; kk < N - M; kk++)
            {
```

```
                z = (mt[kk] & UPPER_MASK) | (mt[kk+1] &
                    LOWER_MASK);
                mt[kk] = mt[kk+M] ^ (z >>> 1) ^ mag01[z
                    & 0x1];
                }
            for (; kk < N-1; kk++)
                {
                z = (mt[kk] & UPPER_MASK) | (mt[kk+1] &
                    LOWER_MASK);
                mt[kk] = mt[kk+(M-N)] ^ (z >>> 1) ^
                    mag01[z & 0x1];
                }
            z = (mt[N-1] & UPPER_MASK) | (mt[0] &
                LOWER_MASK);
            mt[N-1] = mt[M-1] ^ (z >>> 1) ^ mag01[z & 0
                x1];

            mti = 0;
            }

        z = mt[mti++];
        z ^= z >>> 11;                              //
            TEMPERING_SHIFT_U(z)
        z ^= (z << 7) & TEMPERING_MASK_B;           //
            TEMPERING_SHIFT_S(z)
        z ^= (z << 15) & TEMPERING_MASK_C;          //
            TEMPERING_SHIFT_T(z)
        z ^= (z >>> 18);                            //
            TEMPERING_SHIFT_L(z)

        if (mti >= N)    // generate N words at one time
            {
            int kk;
            final int[] mt = this.mt; // locals are
                slightly faster
            final int[] mag01 = this.mag01; // locals
                are slightly faster

            for (kk = 0; kk < N - M; kk++)
                {
                a = (mt[kk] & UPPER_MASK) | (mt[kk+1] &
                    LOWER_MASK);
                mt[kk] = mt[kk+M] ^ (a >>> 1) ^ mag01[a
                    & 0x1];
                }
            for (; kk < N-1; kk++)
```

```
            {
            a = (mt[kk] & UPPER_MASK) | (mt[kk+1] &
                LOWER_MASK);
            mt[kk] = mt[kk+(M-N)] ^ (a >>> 1) ^
                mag01[a & 0x1];
            }
        a = (mt[N-1] & UPPER_MASK) | (mt[0] &
            LOWER_MASK);
        mt[N-1] = mt[M-1] ^ (a >>> 1) ^ mag01[a & 0
            x1];

        mti = 0;
        }

    a = mt[mti++];
    a ^= a >>> 11;                                  //
        TEMPERING_SHIFT_U(a)
    a ^= (a << 7) & TEMPERING_MASK_B;               //
        TEMPERING_SHIFT_S(a)
    a ^= (a << 15) & TEMPERING_MASK_C;              //
        TEMPERING_SHIFT_T(a)
    a ^= (a >>> 18);                                //
        TEMPERING_SHIFT_L(a)

    if (mti >= N)    // generate N words at one time
        {
        int kk;
        final int[] mt = this.mt; // locals are
            slightly faster
        final int[] mag01 = this.mag01; // locals
            are slightly faster

        for (kk = 0; kk < N - M; kk++)
            {
            b = (mt[kk] & UPPER_MASK) | (mt[kk+1] &
                LOWER_MASK);
            mt[kk] = mt[kk+M] ^ (b >>> 1) ^ mag01[b
                & 0x1];
            }
        for (; kk < N-1; kk++)
            {
            b = (mt[kk] & UPPER_MASK) | (mt[kk+1] &
                LOWER_MASK);
            mt[kk] = mt[kk+(M-N)] ^ (b >>> 1) ^
                mag01[b & 0x1];
            }
```

```
                b = (mt[N-1] & UPPER_MASK) | (mt[0] &
                    LOWER_MASK);
                mt[N-1] = mt[M-1] ^ (b >>> 1) ^ mag01[b & 0
                    x1];

                mti = 0;
                }

            b = mt[mti++];
            b ^= b >>> 11;                              //
                TEMPERING_SHIFT_U(b)
            b ^= (b << 7) & TEMPERING_MASK_B;           //
                TEMPERING_SHIFT_S(b)
            b ^= (b << 15) & TEMPERING_MASK_C;          //
                TEMPERING_SHIFT_T(b)
            b ^= (b >>> 18);                            //
                TEMPERING_SHIFT_L(b)

            /* derived from nextDouble documentation in jdk
                1.2 docs, see top */
            v1 = 2 *
                (((((long)(y >>> 6)) << 27) + (z >>> 5)) /
                    (double)(1L << 53))
                - 1;
            v2 = 2 * (((((long)(a >>> 6)) << 27) + (b >>>
                5)) / (double)(1L << 53))
                - 1;
            s = v1 * v1 + v2 * v2;
            } while (s >= 1 || s==0);
        double multiplier = StrictMath.sqrt(-2 * StrictMath
            .log(s)/s);
        __nextNextGaussian = v2 * multiplier;
        __haveNextNextGaussian = true;
        return v1 * multiplier;
        }
    }




/** Returns a random float in the half-open range from [0.0
    f,1.0f).  Thus 0.0f is a valid
    result but 1.0f is not. */
public final float nextFloat()
    {
```

```
    int y;

    if (mti >= N)    // generate N words at one time
        {
        int kk;
        final int[] mt = this.mt; // locals are slightly
            faster
        final int[] mag01 = this.mag01; // locals are
            slightly faster

        for (kk = 0; kk < N - M; kk++)
            {
            y = (mt[kk] & UPPER_MASK) | (mt[kk+1] &
                LOWER_MASK);
            mt[kk] = mt[kk+M] ^ (y >>> 1) ^ mag01[y & 0x1];
            }
        for (; kk < N-1; kk++)
            {
            y = (mt[kk] & UPPER_MASK) | (mt[kk+1] &
                LOWER_MASK);
            mt[kk] = mt[kk+(M-N)] ^ (y >>> 1) ^ mag01[y & 0
                x1];
            }
        y = (mt[N-1] & UPPER_MASK) | (mt[0] & LOWER_MASK);
        mt[N-1] = mt[M-1] ^ (y >>> 1) ^ mag01[y & 0x1];

        mti = 0;
        }

    y = mt[mti++];
    y ^= y >>> 11;                              //
        TEMPERING_SHIFT_U(y)
    y ^= (y << 7) & TEMPERING_MASK_B;           //
        TEMPERING_SHIFT_S(y)
    y ^= (y << 15) & TEMPERING_MASK_C;          //
        TEMPERING_SHIFT_T(y)
    y ^= (y >>> 18);                            //
        TEMPERING_SHIFT_L(y)

    return (y >>> 8) / ((float)(1 << 24));
    }


/** Returns an integer drawn uniformly from 0 to n-1.
    Suffice it to say,
```

```
    n must be > 0, or an IllegalArgumentException is raised
       . */
  public final int nextInt(final int n)
    {
    if (n<=0)
        throw new IllegalArgumentException("n must be
           positive, got: " + n);

    if ((n & -n) == n)  // i.e., n is a power of 2
        {
        int y;

        if (mti >= N)   // generate N words at one time
            {
            int kk;
            final int[] mt = this.mt; // locals are
               slightly faster
            final int[] mag01 = this.mag01; // locals are
               slightly faster

            for (kk = 0; kk < N - M; kk++)
                {
                y = (mt[kk] & UPPER_MASK) | (mt[kk+1] &
                   LOWER_MASK);
                mt[kk] = mt[kk+M] ^ (y >>> 1) ^ mag01[y & 0
                   x1];
                }
            for (; kk < N-1; kk++)
                {
                y = (mt[kk] & UPPER_MASK) | (mt[kk+1] &
                   LOWER_MASK);
                mt[kk] = mt[kk+(M-N)] ^ (y >>> 1) ^ mag01[y
                    & 0x1];
                }
            y = (mt[N-1] & UPPER_MASK) | (mt[0] &
               LOWER_MASK);
            mt[N-1] = mt[M-1] ^ (y >>> 1) ^ mag01[y & 0x1];

            mti = 0;
            }

        y = mt[mti++];
        y ^= y >>> 11;                          //
           TEMPERING_SHIFT_U(y)
        y ^= (y << 7) & TEMPERING_MASK_B;       //
           TEMPERING_SHIFT_S(y)
```

```
        y ^= (y << 15) & TEMPERING_MASK_C;          //
            TEMPERING_SHIFT_T(y)
        y ^= (y >>> 18);                            //
            TEMPERING_SHIFT_L(y)

        return (int)((n * (long) (y >>> 1) ) >> 31);
        }

    int bits, val;
    do
        {
        int y;

        if (mti >= N)    // generate N words at one time
            {
            int kk;
            final int[] mt = this.mt; // locals are
                slightly faster
            final int[] mag01 = this.mag01; // locals are
                slightly faster

            for (kk = 0; kk < N - M; kk++)
                {
                y = (mt[kk] & UPPER_MASK) | (mt[kk+1] &
                    LOWER_MASK);
                mt[kk] = mt[kk+M] ^ (y >>> 1) ^ mag01[y & 0
                    x1];
                }
            for (; kk < N-1; kk++)
                {
                y = (mt[kk] & UPPER_MASK) | (mt[kk+1] &
                    LOWER_MASK);
                mt[kk] = mt[kk+(M-N)] ^ (y >>> 1) ^ mag01[y
                     & 0x1];
                }
            y = (mt[N-1] & UPPER_MASK) | (mt[0] &
                LOWER_MASK);
            mt[N-1] = mt[M-1] ^ (y >>> 1) ^ mag01[y & 0x1];

            mti = 0;
            }

        y = mt[mti++];
        y ^= y >>> 11;                              //
            TEMPERING_SHIFT_U(y)
```

```
        y ^= (y << 7) & TEMPERING_MASK_B;        //
            TEMPERING_SHIFT_S(y)
        y ^= (y << 15) & TEMPERING_MASK_C;       //
            TEMPERING_SHIFT_T(y)
        y ^= (y >>> 18);                         //
            TEMPERING_SHIFT_L(y)

        bits = (y >>> 1);
        val = bits % n;
        } while(bits - val + (n-1) < 0);
    return val;
    }


/**
 * Tests the code.
 */
public static void main(String args[])
    {
    int j;

    MersenneTwisterFast r;

    // CORRECTNESS TEST
    // COMPARE WITH http://www.math.keio.ac.jp/matumoto/
       CODES/MT2002/mt19937ar.out

    r = new MersenneTwisterFast(new int[]{0x123, 0x234, 0
       x345, 0x456});
    System.out.println("Output␣of␣MersenneTwisterFast␣with␣
       new␣(2002/1/26)␣seeding␣mechanism");
    for (j=0;j<1000;j++)
        {
        // first, convert the int from signed to "unsigned"
        long l = (long)r.nextInt();
        if (l < 0 ) l += 4294967296L;  // max int value
        String s = String.valueOf(l);
        while(s.length() < 10) s = "␣" + s;  // buffer
        System.out.print(s + "␣");
        if (j%5==4) System.out.println();
        }

    // SPEED TEST

    final long SEED = 4357;
```

```
int xx; long ms;
System.out.println("\nTime␣to␣test␣grabbing␣100000000␣
    ints");

Random rr = new Random(SEED);
xx = 0;
ms = System.currentTimeMillis();
for (j = 0; j < 100000000; j++)
     xx += rr.nextInt();
System.out.println("java.util.Random:␣" + (System.
    currentTimeMillis()-ms) + "␣␣␣␣␣␣␣␣␣␣Ignore␣this:␣"
    + xx);

r = new MersenneTwisterFast(SEED);
ms = System.currentTimeMillis();
xx=0;
for (j = 0; j < 100000000; j++)
     xx += r.nextInt();
System.out.println("Mersenne␣Twister␣Fast:␣" + (System.
    currentTimeMillis()-ms) + "␣␣␣␣␣␣␣␣␣␣Ignore␣this:␣"
    + xx);

// TEST TO COMPARE TYPE CONVERSION BETWEEN
// MersenneTwisterFast.java AND MersenneTwister.java

System.out.println("\nGrab␣the␣first␣1000␣booleans");
r = new MersenneTwisterFast(SEED);
for (j = 0; j < 1000; j++)
     {
     System.out.print(r.nextBoolean() + "␣");
     if (j%8==7) System.out.println();
     }
if (!(j%8==7)) System.out.println();

System.out.println("\nGrab␣1000␣booleans␣of␣increasing␣
    probability␣using␣nextBoolean(double)");
r = new MersenneTwisterFast(SEED);
for (j = 0; j < 1000; j++)
     {
     System.out.print(r.nextBoolean((double)(j/999.0)) +
          "␣");
     if (j%8==7) System.out.println();
     }
if (!(j%8==7)) System.out.println();
```

```
System.out.println("\nGrab␣1000␣booleans␣of␣increasing␣
    probability␣using␣nextBoolean(float)");
r = new MersenneTwisterFast(SEED);
for (j = 0; j < 1000; j++)
    {
    System.out.print(r.nextBoolean((float)(j/999.0f)) +
        "␣");
    if (j%8==7) System.out.println();
    }
if (!(j%8==7)) System.out.println();

byte[] bytes = new byte[1000];
System.out.println("\nGrab␣the␣first␣1000␣bytes␣using␣
    nextBytes");
r = new MersenneTwisterFast(SEED);
r.nextBytes(bytes);
for (j = 0; j < 1000; j++)
    {
    System.out.print(bytes[j] + "␣");
    if (j%16==15) System.out.println();
    }
if (!(j%16==15)) System.out.println();

byte b;
System.out.println("\nGrab␣the␣first␣1000␣bytes␣--␣must
    ␣be␣same␣as␣nextBytes");
r = new MersenneTwisterFast(SEED);
for (j = 0; j < 1000; j++)
    {
    System.out.print((b = r.nextByte()) + "␣");
    if (b!=bytes[j]) System.out.print("BAD␣");
    if (j%16==15) System.out.println();
    }
if (!(j%16==15)) System.out.println();

System.out.println("\nGrab␣the␣first␣1000␣shorts");
r = new MersenneTwisterFast(SEED);
for (j = 0; j < 1000; j++)
    {
    System.out.print(r.nextShort() + "␣");
    if (j%8==7) System.out.println();
    }
if (!(j%8==7)) System.out.println();

System.out.println("\nGrab␣the␣first␣1000␣ints");
r = new MersenneTwisterFast(SEED);
```

```
for (j = 0; j < 1000; j++)
    {
    System.out.print(r.nextInt() + " ");
    if (j%4==3) System.out.println();
    }
if (!(j%4==3)) System.out.println();

System.out.println("\nGrab the first 1000 ints of
   different sizes");
r = new MersenneTwisterFast(SEED);
int max = 1;
for (j = 0; j < 1000; j++)
    {
    System.out.print(r.nextInt(max) + " ");
    max *= 2;
    if (max <= 0) max = 1;
    if (j%4==3) System.out.println();
    }
if (!(j%4==3)) System.out.println();

System.out.println("\nGrab the first 1000 longs");
r = new MersenneTwisterFast(SEED);
for (j = 0; j < 1000; j++)
    {
    System.out.print(r.nextLong() + " ");
    if (j%3==2) System.out.println();
    }
if (!(j%3==2)) System.out.println();

System.out.println("\nGrab the first 1000 longs of
   different sizes");
r = new MersenneTwisterFast(SEED);
long max2 = 1;
for (j = 0; j < 1000; j++)
    {
    System.out.print(r.nextLong(max2) + " ");
    max2 *= 2;
    if (max2 <= 0) max2 = 1;
    if (j%4==3) System.out.println();
    }
if (!(j%4==3)) System.out.println();

System.out.println("\nGrab the first 1000 floats");
r = new MersenneTwisterFast(SEED);
for (j = 0; j < 1000; j++)
    {
```

```
            System.out.print(r.nextFloat() + " ");
            if (j%4==3) System.out.println();
            }
        if (!(j%4==3)) System.out.println();

        System.out.println("\nGrab the first 1000 doubles");
        r = new MersenneTwisterFast(SEED);
        for (j = 0; j < 1000; j++)
            {
            System.out.print(r.nextDouble() + " ");
            if (j%3==2) System.out.println();
            }
        if (!(j%3==2)) System.out.println();

        System.out.println("\nGrab the first 1000 gaussian
            doubles");
        r = new MersenneTwisterFast(SEED);
        for (j = 0; j < 1000; j++)
            {
            System.out.print(r.nextGaussian() + " ");
            if (j%3==2) System.out.println();
            }
        if (!(j%3==2)) System.out.println();

        }
    }
```

FELW2ST.java

```
/*
 Copyright 2013 by Tom Wilkinson and Cardiff University
 Licensed under the Academic Free License version 3.0
 See the file "LICENSE" for more information
*/
package util;
import edu.emory.mathcs.jtransforms.fft.DoubleFFT_1D;


/** A transcription of the MATLAB implementation (by Katsumi
    Shimotsu) of the
* Feasible Exact Local Whittle 2-STage estimator described in
    Simotsu (2010)
*
* WARNING: - not compatible with complex-number data
*
```

```
* This version uses quasi-Netwon updating instead of Newton
    steps.
*       It produces more stable estimates than the previous
    version when m is small.
*
*       This code uses a modified version of Chris Sims' BFGS
    optimization software,
*           specifically bfgsi_ks.m, csminit_ks.m, csminwel_ks.
    m, and numgrad.m.
*
*       Visit Chris Sims' webpage http://www.princeton.edu/~
    sims for the original code
*       and its documentation. I thank Chris Sims for
    generously making his Matlab codes
*        publicly available.
*
* 24/04/2013 - output compared to MATLAB code output for
    dataset from Mayoral, 2006
* (Oxford Bulletin of Economics and Statistics, 68)
* Outcome: identical output
*/
public class FELW2St {
  static double[] data;
  static int numFreq;
  static int taperOrder;

  public FELW2St(double[] data, int numFreq, int taperOrder){
      this.data = data;
      this.numFreq = numFreq;
      this.taperOrder = taperOrder;
  }

  private static class Gradient{
      double gradient;
      boolean badGradient;

      Gradient(double gradient, boolean badGradient){
          this.gradient = gradient;
          this.badGradient = badGradient;
      }
   }

  private static class CSMinPass{
      double whittleVal;
      double parVal;
      int whittleCount;
```

```
    int returnCode;
    /* 0 => normal step; 1 => zero gradient; 2, 4 => back
       and forth adjustment
     * of stepsize didn't finish; 3 => smallest stepsize
        still improves too
     * slow; 5 => largest stepsize still improves too fast;
        6 => no
     * improvement found */

    CSMinPass(double whittleVal, double parVal, int
       whittleCount, int returnCode){
        this.whittleVal = whittleVal;
        this.parVal = parVal;
        this.whittleCount = whittleCount;
        this.returnCode = returnCode;
    }
 }

public double estimate(){
    /* first minimise Velasco's tapered local Whittle
       likelihood */
    double initialPar = fminbnd(-1,3);
    double convergenceCriterion = Math.pow(10.0,-12.0);
    int maxIterations = 100;

    /* now minimise the exact Whittle likelihood */
    return csminwel_ks(initialPar,4,convergenceCriterion,
       maxIterations,true);
}


/* this is a straight up renaming of Steve Verrill's
   translation of the
 * FORTRAN Fmin. His header:
 *This class was translated by a statistician from the
    FORTRAN
 *version of fmin.  It is NOT an official translation.  When
 *public domain Java optimization routines become available
 *from professional numerical analysts, then <b>THE CODE
    PRODUCED
 *BY THE NUMERICAL ANALYSTS SHOULD BE USED</b>.
 *
 *<p>
 *Meanwhile, if you have suggestions for improving this
 *code, please contact Steve Verrill at steve@www1.fpl.fs.
    fed.us.
```

```
*
*@author Steve Verrill
*@version .5 --- March 24, 1998 */
static double fminbnd(double parMin, double parMax){

    double goldenRatio = .5*(3.0 - Math.sqrt(5.0));
    double increment = 0.0;

    // 1.1102e-16 is machine precision

    double miniscule = Math.sqrt(1.2e-16);

    double olderPar = parMin + goldenRatio*(parMax-parMin);
    double oldPar = olderPar;
    double diffPar = olderPar;
    double furthestFromBound = 0.0;
    double whittle = veltaper(diffPar);
    double olderWhittle = whittle;
    double oldWhittle = whittle;
    double tolerance3 = Math.pow(10.0,-12.0)/3.0;

    double parMean = 0.5*(parMin + parMax);
    double tolerance1 = miniscule*Math.abs(diffPar) +
       tolerance3;
    double tolerance2 = 2.0*tolerance1;

    // main loop
    while (Math.abs(diffPar-parMean) > (tolerance2 - 0.5*(
       parMax-parMin))){
        double p = 0.0;
        double q = 0.0;
        double r = 0.0;
        if (Math.abs(furthestFromBound) > tolerance1) {
            // fit the parabolae
            r = (diffPar-oldPar)*(whittle-olderWhittle);
            q = (diffPar-olderPar)*(whittle-oldWhittle);

            p = (diffPar-olderPar)*q - (diffPar-oldPar)*r;
            q = 2.0*(q-r);

            if (q > 0.0) {
                p = -p;
            }
            else {
                q = -q;
            }
```

```
        r = furthestFromBound;
        furthestFromBound = increment;

        // brace below corresponds to statement 50
    }

    if ((Math.abs(p) < Math.abs(.5*q*r))
            && (p > q*(parMin-diffPar))
            && (p < q*(parMax-diffPar))) {

        // a parabolic interpolation step
        increment = p/q;
        double newPar = diffPar+increment;

        // veltaper must not be evaluated too close to
           parMin or parMax
        if (((newPar-parMin) < tolerance2) || ((parMax-
           newPar) < tolerance2)) {
            increment = tolerance1;
            if (diffPar >= parMean) increment = -
               increment;
        }
    // brace below corresponds to statement 60
    }
    else {
        // a golden-section step
        if (diffPar < parMean) {
            furthestFromBound = parMax-diffPar;
        }
        else {
            furthestFromBound = parMin-diffPar;
        }
        increment = goldenRatio*furthestFromBound;

    }

    // veltaper must not be evaluated too close to
       diffPar
    double newPar = 0.0;
    if (Math.abs(increment) >= tolerance1) {
        newPar = diffPar+increment;
    }
    else {
        if (increment > 0.0) {
            newPar = diffPar + tolerance1;
        }
```

```
        else {
            newPar = diffPar - tolerance1;
        }
    }

    double newWhittle = veltaper(newPar);

    // Update parMin, parMax, v, w, and diffPar
    if (whittle <= newWhittle) {
        /* no improvement means narrowing the search to
            diffPar's side of
         * newPar */
        if (newPar < diffPar) {
            parMin = newPar;
        }
        else {
            parMax = newPar;
        }
        // brace below corresponds to statement 140
    }

    if (newWhittle <= whittle) {
        /* an improvement means narrowing search to
           newPar's side of
         * diffPar */
        if (newPar < diffPar) {
            parMax = diffPar;
        }
        else {
            parMin = diffPar;
        }

        olderPar = oldPar;
        olderWhittle = oldWhittle;
        oldPar = diffPar;
        oldWhittle = whittle;
        diffPar = newPar;
        whittle = newWhittle;

        /* reset the following for the new parMin,
           parMax and diffPar */
        parMean = 0.5*(parMin + parMax);
        tolerance1 = miniscule*Math.abs(diffPar) +
           tolerance3;
        tolerance2 = 2.0*tolerance1;
```

```
            // brace below corresponds to statement 170
        }
        else {
            /* no improvement means replacing one of the
               other reference
             * points */
            if ((newWhittle <= oldWhittle) || (oldPar ==
               diffPar)) {
                /* the more recent reference value is
                   beaten, and replaced */
                olderPar = oldPar;
                olderWhittle = oldWhittle;
                oldPar = newPar;
                oldWhittle = newWhittle;

                parMean = 0.5*(parMin + parMax);
                tolerance1 = miniscule*Math.abs(diffPar) +
                   tolerance3;
                tolerance2 = 2.0*tolerance1;
            }
            else if ((newWhittle > olderWhittle) && (
               olderPar != diffPar)
                    && (olderPar != oldPar)) {
                /* the new value is worse than all
                   reference values! */
                parMean = 0.5*(parMin + parMax);
                tolerance1 = miniscule*Math.abs(diffPar) +
                   tolerance3;
                tolerance2 = 2.0*tolerance1;
            }
            else {
                /* otherwise the less recent reference
                   value is replaced */
                olderPar = newPar;
                olderWhittle = newWhittle;

                parMean = 0.5*(parMin + parMax);
                tolerance1 = miniscule*Math.abs(diffPar) +
                   tolerance3;
                tolerance2 = 2.0*tolerance1;
            }
        }

    // brace below corresponds to statement 190
    }
```

```
        return diffPar;
    }

    /* tested: behaves like MATLAB (for d=0.52, p=3, m=22, data
       from mdata.csv) */
    static double veltaper(double diffPar){
        int dataSize = data.length;

        double[] taperedData = data.clone();
        if(taperOrder == 2){
            double median = Math.ceil((double)dataSize/2.0);
            for(int t=0;t<dataSize;t++)
                taperedData[t] *= (1.0 - Math.abs(t+1.0-median)/
                    median);
        }
        else{
            int convolvingLength = (int)Math.floor((dataSize
                +2.0)/3.0);

            for(int t=0; t<dataSize; t++){
                double tapering = 0.0;
                for(int s=1; s<=convolvingLength; s++){
                    if(t+1 >= s && s > t+1 - convolvingLength)
                        tapering += s;
                    if(t+1-convolvingLength < s+convolvingLength
                            && s + convolvingLength <= t+1)
                        tapering += convolvingLength - s;
                }
                taperedData[t] *= tapering;
            }
        }

        double[] ftTaperedData = new double[2*dataSize];
        for(int i=0; i<dataSize; i++){
            ftTaperedData[2*i] = taperedData[i];
        }
        new DoubleFFT_1D(dataSize).complexForward(ftTaperedData)
            ;
        double frequency = 2.0*Math.PI/(double)dataSize;
        /* The complex data vector now has real components
           followed by imaginary
         * components in sequence; multiplication with exp(i*
           lambda) will have to
         * be done two entries at a time */
        double periodogram = 0.0;
        double logFreqSum = 0.0;
```

```
        for(int k=taperOrder; k<=numFreq-1; k+=taperOrder){
            double thisFreq = frequency*(double)k;
            logFreqSum += Math.log(thisFreq);
            /* first to find the real and imaginary parts of
             * conj(fft(conj(diffData)))*exp(i*frequency)... */
            double realPart = ftTaperedData[2*k] * Math.cos(
                thisFreq)
                    + ftTaperedData[2*k+1] * Math.sin(thisFreq);
                        // -i*i = 1
            double imaginaryPart = ftTaperedData[2*k] * Math.sin
                (thisFreq)
                    - ftTaperedData[2*k+1] * Math.cos(thisFreq);
                        // -i * 1 = -i
            /* now to add to the periodogram wdx*conj(wdx) = re*
                re + im*im */
            periodogram += Math.pow(thisFreq,2.0*diffPar)
                    *(realPart*realPart + imaginaryPart*
                        imaginaryPart);
        }
        periodogram *= 1.0/(2.0*Math.PI*(double)dataSize);
        periodogram *= (double)taperOrder/(double)numFreq;

        /* return the Whittle likelihood */
        return Math.log(periodogram)
                - 2.0*diffPar*logFreqSum *(double)taperOrder /(
                    double)numFreq;
    }

    static double csminwel_ks(double initialPar, double
        initialHess,
            double convergenceCriterion, int maxIterations,
                boolean estimateFirstDatum){
        boolean done = false;
        int retcode = 0;
        int iterations = 0;
        int whittleCount = 0;
        int snit = 100;

        /* get initial exact whittle estimate */
        double initialWhittle = ewhittle(initialPar,
            estimateFirstDatum);

        /* check workable initial parameter has been passed */
        if(initialWhittle > Math.pow(10.0,50.0))
            throw new RuntimeException("Bad initial parameter.")
                ;
```

```
/* now numerical gradient estimation */
Gradient gradient = numgrad(initialPar,
   estimateFirstDatum);
retcode = 101;

/* the iterations begin */
double parVal = initialPar;
double whittleVal = initialWhittle;
double hessVal = initialHess;
boolean badGradient = false;

while(!done){
    iterations++;
    CSMinPass cSMinPass1
            = csminit_ks(parVal, whittleVal, gradient,
                hessVal, estimateFirstDatum);
    whittleCount += cSMinPass1.whittleCount;
    /* if the gradient is not effectively zero we need
       its new value... */
    Gradient gradient1 = new Gradient(0.0,true);
    if(cSMinPass1.returnCode != 1){
        /* unless the back and forth adjustment of
           stepsize didn't finish */
        if(cSMinPass1.returnCode == 2 || cSMinPass1.
           returnCode == 4){
            gradient1.badGradient = true;
        }
        else gradient1 = numgrad(cSMinPass1.parVal,
           estimateFirstDatum);
    }
    /* if it is effectively zero, then the whittle
       likelihood can't have
     * changed */
    else cSMinPass1.whittleVal = whittleVal;

    CSMinPass cSMinPassh
            = new CSMinPass(whittleVal, parVal,
                    cSMinPass1.whittleCount, cSMinPass1.
                       returnCode);
    Gradient gradienth;

    /* if the new whittle value improves on the previous
        one, and the
     * gradient is not bad, then  */
    double whittleReduction = whittleVal - cSMinPass1.
       whittleVal;
```

```
            boolean stuck = (Math.abs(whittleReduction)<
                convergenceCriterion);

            if(convergenceCriterion < whittleReduction
                    && gradient1.badGradient == false) {
                /* if the gradient is not effectively zero, then
                    the whittle
                 * likelihood must be updated... */
                cSMinPassh = cSMinPass1;
                gradienth = gradient1;
            }
            else {
                if(cSMinPass1.whittleVal < whittleVal)
                   cSMinPassh=cSMinPass1;
                gradienth = numgrad(parVal,estimateFirstDatum);
                gradienth.badGradient = true;
            }
            if(!gradient.badGradient && !gradienth.badGradient
               && !stuck)
                hessVal = bfgsi_ks(hessVal,
                        gradienth.gradient - gradient.gradient,
                        cSMinPassh.parVal-parVal);

            whittleVal = cSMinPassh.whittleVal;
            parVal = cSMinPassh.parVal;
            gradient = gradienth;

            /* we can stop if the reduction is less than the
               convergence criterion
             * OR if we have exceeded the allowed iterations */
            if(stuck || (iterations>maxIterations)) done = true;
        }
        return parVal;
    }

    /* tested: behaves like MATLAB (for d=0.52, p=3, m=22, data
       from mdata.csv) */
    static double ewhittle(double diffPar, boolean
       estimateFirstDatum){

        double[] taperedData = data.clone();
        int dataSize = data.length;
        if(estimateFirstDatum){
            double weight = ((diffPar <= 0.5)?1.0:0.0)
                + 0.5*(1.0 + Math.cos(-2.0*Math.PI + 4.0*Math.PI
                    *diffPar))
```

```
            *((diffPar>0.5)?1.0:0.0)*((diffPar<0.75)
                ?1.0:0.0);
        double dataSum = 0.0;
        for(int i=0; i<dataSize; i++){
            dataSum+=data[i];
        }
        double myu = weight*dataSum/(double)dataSize + (1.0-
            weight)*data[0];

        for(int i=0; i<dataSize; i++){
            taperedData[i] = data[i] - myu;
        }
    }

    double[] diffData = fracdiff(taperedData,diffPar);
    double[] ftDiffData = new double[2*dataSize];
    for(int i=0; i<dataSize; i++){
        ftDiffData[2*i] = diffData[i];
    }
    new DoubleFFT_1D(dataSize).complexForward(ftDiffData);
    double frequency = 2.0*Math.PI/(double)dataSize;
    /* The complex data vector now has real components
       followed by imaginary
     * components in sequence; multiplication with exp(i*
       lambda) will have to
     * be done two entries at a time */
    double periodogram = 0.0;
    double logFreqSum = 0.0;
    /* we want to drop the first observation, */
    for(int k=1; k<numFreq+1; k++){
        double thisFreq = frequency*(double)k;
        logFreqSum += Math.log(thisFreq);
        /* first to find the real and imaginary parts of
         * conj(fft(conj(diffData)))*exp(i*frequency)... */
        double realPart = ftDiffData[2*k] * Math.cos(
            thisFreq)
                + ftDiffData[2*k+1] * Math.sin(thisFreq); //
                    -i*i = 1
        double imaginaryPart = ftDiffData[2*k] * Math.sin(
            thisFreq)
                - ftDiffData[2*k+1] * Math.cos(thisFreq); //
                    -i * 1 = -i
        /* now to add to the periodogram wdx*conj(wdx) = re*
            re + im*im */
        periodogram += realPart*realPart + imaginaryPart*
            imaginaryPart;
```

```
        }
        periodogram *= 1.0/(2.0*Math.PI*(double)dataSize*(double
            )numFreq);

        /* return the Whittle likelihood */
        return Math.log(periodogram) - 2.0*diffPar*logFreqSum /
            (double)numFreq;
    }

    /* tested: behaves like MATLAB (for d=0.52, p=3, m=22, data
       from mdata.csv) */
    public static double[] fracdiff(double[] demeanedData,
        double diffPar){
        int dataSize = demeanedData.length;
        double[] diffData = new double[dataSize];

        double lagMultiplier = 1.0;
        for(int k=0; k<dataSize; k++){
            for(int j=k; j<dataSize; j++){
                diffData[j] += lagMultiplier * demeanedData[j-k
                    ];
            }
            lagMultiplier *= (k-diffPar)/(k+1);
        }

        return diffData;
    }

    static CSMinPass csminit_ks(double initialPar, double
        initialWhittle,
            Gradient initialGradient, double initialHess,
            boolean estimateFirstDatum){
        final double ANGLE = 0.005;
        final double THETA = 0.3;
        final int WHITTLE_CHANGE = 1000;
        final double MIN_STEP = Math.pow(10.0,-9.0);
        final double MIN_FACTOR_CHANGE = 0.01;

        int whittleCount = 0;
        double stepSize = 1.0;
        double parOut = initialPar;
        double whittleVal = initialWhittle;
        double whittleOut = initialWhittle;
        double gradient = initialGradient.gradient;
        boolean badGradient = initialGradient.badGradient;
        double gradientNorm = Math.abs(gradient);
```

```
double parChange = 0.0;
double parChangeNorm = 0.0;

CSMinPass cSMinPass = new CSMinPass(whittleVal,parOut,
    whittleCount,0);
/* already done if the gradient is effectively zero */
if(gradientNorm < Math.pow(10.0,-12.0) && !badGradient){
    cSMinPass.returnCode = 1;
    parChangeNorm = 0.0;
}
/* otherwise, can begin exploring for a minimum */
else {
    parChange = -initialHess*gradient;
    parChangeNorm = Math.abs(parChange);
    if(parChangeNorm > Math.pow(10.0,12.0)){
        /* near singular hessian */
        parChange = parChange*WHITTLE_CHANGE/
            parChangeNorm;
    }
    double whittleOutChange = parChange*initialGradient.
        gradient;
    if(!badGradient){
        /* test for alignment of the change in parameter
            with gradient */
        if(ANGLE > -whittleOutChange/(gradientNorm*
            parChangeNorm)){
            parChange -= gradient*(ANGLE*parChangeNorm/
                gradientNorm
                    + whittleOutChange / (gradientNorm*
                        gradientNorm));
            /* to keep the scale invariant to the angle
                correction... */
            parChange *= parChangeNorm/Math.abs(
                parChange);
            whittleOutChange = parChange*gradient;
        }
    }
    /* now adjust length of step until min and max
        improvement criteria
     * are met */
    boolean done = false;
    double factor = 3.0;
    boolean shrink = true;
    double stepMin = 0.0;
    double stepMax = Double.POSITIVE_INFINITY;
    double stepPeak = 0.0;
```

```
double whittlePeak = initialWhittle;
double stepOut = 0.0;
while(!done){
    double testPar = initialPar+parChange*stepSize;
    whittleVal = ewhittle(testPar,estimateFirstDatum
        );

    if(whittleVal<whittleOut){
        whittleOut = whittleVal;
        parOut = testPar;
        stepOut = stepSize;
    }
    whittleCount++;
    boolean shrinkSignal = (!badGradient
            && (initialWhittle-whittleVal
            < Math.max(-THETA*whittleOutChange*
                stepSize,0.0)))
            || (badGradient && (initialWhittle-
                whittleVal < 0.0));
    boolean growSignal = !badGradient && ((stepSize
        >0.0)
            && (initialWhittle - whittleVal
            > -(1-THETA)*whittleOutChange*stepSize))
                ;

    if(shrinkSignal && ((stepSize>stepPeak)||(
        stepSize<0.0))){
        if( (stepSize > 0.0)
                && (!shrink || (stepSize/factor <=
                    stepPeak))){
            shrink = true;
            factor = Math.pow(factor,0.6);
            while(stepSize/factor <= stepPeak)
                factor = Math.pow(factor,0.6);

            if(Math.abs(factor-1)<MIN_FACTOR_CHANGE)
                {
                if(Math.abs(stepSize)<4.0) cSMinPass
                    .returnCode = 2;
                else cSMinPass.returnCode = 7;
                done = true;
            }
        }
        if((stepSize<stepMax) && (stepSize>stepPeak)
            ) stepMax = stepSize;
        stepSize = stepSize/factor;
```

```
            if(Math.abs(stepSize)<MIN_STEP){
                /* we try going against the gradient,
                   which may be
                 * inaccurate, if there was no gain in
                   likelihood */
                if((stepSize > 0.0) && (initialWhittle
                   <= whittleOut))
                    stepSize = -stepSize*Math.pow(factor
                       ,0.6);
                else {
                    if (stepSize < 0.0) cSMinPass.
                       returnCode = 6;
                    else cSMinPass.returnCode = 3;
                    done = true;
                }
            }
        }
        else if (growSignal && (stepSize > 0.0)
                || (shrinkSignal && ((stepSize <=
                   stepPeak)&&(stepSize >0.0)))){
            if(shrink){
                    shrink = false;
                    factor = Math.pow(factor, 0.6);
                    if(Math.abs(factor - 1)<
                       MIN_FACTOR_CHANGE){
                        if(Math.abs(stepSize)<4.0)
                           cSMinPass.returnCode = 4;
                        else cSMinPass.returnCode = 7;
                        done = true;
                    }
            }
            if((whittleVal < whittlePeak) && (stepSize
               >0.0)){
                whittlePeak = whittleVal;
                stepPeak = stepSize;
                if(stepMax<=stepPeak) stepMax = stepPeak
                   *factor*factor;
            }
            stepSize *= factor;
            if(Math.abs(stepSize)>Math.pow(10.0, 20.0)){
                cSMinPass.returnCode = 5;
                done = true;
            }
        }
        else{
            done = true;
```

```
                    if(factor < 1.2) cSMinPass.returnCode = 7;
                    else cSMinPass.returnCode = 0;
                }
            }
        }
        cSMinPass.parVal = parOut;
        cSMinPass.whittleVal = whittleOut;
        cSMinPass.whittleCount = whittleCount;
        return cSMinPass;
    }

    static Gradient numgrad(double parVal, boolean
        estimateFirstDatum){
        Gradient gradient = new Gradient(0.0,false);

        double infinitesimal = Math.pow(10.0,-6.0);

        double initialWhittle = ewhittle(parVal,
            estimateFirstDatum);

        double gradValue = (ewhittle(parVal+infinitesimal,
            estimateFirstDatum)
                - initialWhittle) / infinitesimal;

        if(Math.abs(gradValue)< Math.pow(10.0,15.0)) gradient.
            gradient = gradValue;
        else { gradient.gradient = 0.0; gradient.badGradient =
            true; }

        return gradient;
    }

    static double bfgsi_ks(double initialHess, double
        gradientChange,
            double parChange){
        double hessOut;
        /* a slight nomenclature change to allow names for some
            convenient terms:
         * "d" prefix denotes change, "x" denotes multiplication
            */
        double hessxdGrad = initialHess*gradientChange;
        double dGradxdPar = gradientChange*parChange;
        /* providing the denominator won't cause problems */
        if(Math.abs(dGradxdPar) > 1.0e-12)
            hessOut = initialHess
```

```
                    + (parChange + hessxdGrad - 2*hessxdGrad)/
                        gradientChange;
        else hessOut = initialHess;

        return hessOut;
    }
}
```

3.A.4 The alternative model for the robustness check

This is identical to the simpler model, except for the main methods of the *Hierarchy* object:

HierarchySwap.java

```
/*
 Copyright 2013 by Tom Wilkinson and Cardiff University
 Licensed under the Academic Free License version 3.0
 See the file "LICENSE" for more information
*/
package enforcers;
import util.*;

/**This class contains the methods which represent events
    affecting the
* enforcement hierarchy:
* - a network growth event
* - a network collapse event
* - a measuring of productive activity
*
* TERMINOLOGY (for sake of readability): enforcer = sovereign
    ; enforcee = subject;
* domain = all subjects of a sovereign, all those subjects'
    subjects etc ;
* superSovereigns = all those other agents who include the
    specific agent in
* their domains */
public class HierarchySwap {
    int numAgents;
    int maxSubjects;
    double contagionProb;

    MersenneTwisterFast random;
    Bag components;
```

```
Bag agents;
int[] domainDistribution;

/* we'll use a directed network, to capture the directed
   nature of
 * enforcement */
public HierarchySwap(int numAgents, int maxSubjects, double
    contagionProb){
    /* directed by default */
    super();
    this.numAgents = numAgents;
    this.maxSubjects = maxSubjects;
    this.contagionProb = contagionProb;

    random = new MersenneTwisterFast(System.
       currentTimeMillis());
    agents = new Bag();
    domainDistribution = new int[numAgents];
}

/* Agents without are offered a new sovereign */
public void subjugate(){
    /* A new node looks for an enforcer */
    EnforceAgent newSubject
            = (EnforceAgent)agents.objs[random.nextInt(
               numAgents)];

    EnforceAgent sovereign =
                (EnforceAgent)agents.objs[random.nextInt(
                   numAgents)];
    /* there is guaranteed to be some sovereign out there
       who doesn't
     * have full degree, by virtue of the tree structure
       having leaves */
    while(sovereign.equals(newSubject)||!(sovereign.
       numSubjects<maxSubjects)
            ||sovereign.superSovereigns.contains(newSubject
               )){
        sovereign = (EnforceAgent)agents.objs[random.
           nextInt(numAgents)];
    }
    /* this sovereign will be chosen over an existing
       sovereign if it means
     * having a better component, but only relative to the
       number of supersovereigns:
```

```
 * - in the real world, the bigger a component the less
     likely its high ups
 *   are to leave for another: the more their proximity
     to the top matters
 * - to reproduce this, the comparison of component vs
    supersovereigns has
 *   to give component size decreasing weight
 * numAgents MUST NOT BE GREATER THAN 40000, OR
    CONVERSION TO DOUBLE IS
 * NEEDED TO PREVENT INTEGER WRAP-AROUND */
Boolean sameComponent = sovereign.component.contains(
   newSubject);
if((sameComponent
        &&(newSubject.superSovereigns.numObjs
        >sovereign.superSovereigns.numObjs+1))
        ||((!sameComponent)
        &&(sovereign.component.numObjs+newSubject.
            domain.numObjs
        *fastIntPow(maxSubjects,newSubject.
            superSovereigns.numObjs)
        >(newSubject.component.numObjs-1)
        *fastIntPow(maxSubjects,sovereign.
            superSovereigns.numObjs+1)))){
    sovereign.subjects.add(newSubject);

    /* every super-sovereign of the subject must lose
       the subject's
     * domain and the subject from their own */
    Bag oldSuperSovereigns = newSubject.superSovereigns
       ;
    Bag lostDomain = new Bag(newSubject.domain);
    lostDomain.add(newSubject);
    int numOldSupers = oldSuperSovereigns.numObjs;
    for(int superIndex=0; superIndex<numOldSupers;
       superIndex++){
        EnforceAgent oldSuper = (EnforceAgent)(
            oldSuperSovereigns.objs[superIndex]);
        oldSuper.domain.removeAll(lostDomain);
        /* the old sovereign has to be found and reset
           */
        if(oldSuper.subjects.contains(newSubject)){
            oldSuper.subjects.remove(newSubject);
            oldSuper.numSubjects--;
        }
    }
```

```
/* every member of the subject's domain must lose
   the subject's
 * super-sovereigns */
Bag oldDomain = newSubject.domain;
Bag lostSupers = new Bag(newSubject.superSovereigns
   );
int numOldDomain = oldDomain.numObjs;
for(int domainIndex=0; domainIndex<numOldDomain;
   domainIndex++){
    ((EnforceAgent)oldDomain.objs[domainIndex]).
       superSovereigns
            .removeAll(lostSupers);
}
/* the newSubject needs to have the supers bag
   emptied too */
newSubject.superSovereigns = new Bag();

sovereign.numSubjects++;
/* The sovereign's domain, along with all their
   super-sovereigns'
 * domains, will now also include the newSubject
    and all their
 * existing domain */
sovereign.domain.add(newSubject);
sovereign.domain.addAll(newSubject.domain);
Bag superSovereigns = sovereign.superSovereigns;
int numSupers = superSovereigns.numObjs;
for(int superIndex=0; superIndex<numSupers;
   superIndex++){
    Bag superDomain = ((EnforceAgent)
       superSovereigns.objs[superIndex])
            .domain;
    superDomain.add(newSubject);
    superDomain.addAll(newSubject.domain);
}
/* meanwhile, the newSubject will now share all the
    sovereign's
 * super-sovereigns */
newSubject.superSovereigns.add(sovereign);
newSubject.superSovereigns.addAll(sovereign.
   superSovereigns);
/* As should every member of their domain! */
Bag subDomain = newSubject.domain;
int numSubjects = subDomain.size();
for(int subjectIndex=0; subjectIndex<numSubjects;
   subjectIndex++){
```

```
                Bag subSuperSovereigns =
                        ((EnforceAgent)subDomain.objs[
                            subjectIndex])
                        .superSovereigns;
                if(!subSuperSovereigns.contains(sovereign))
                    subSuperSovereigns.add(sovereign);
                subSuperSovereigns.addAll(sovereign.
                    superSovereigns);
            }
        }
    }

    public void sovereignFail(){
        Bag failures = new Bag();

        /* randomly choose an agent to fail - though they may
           have no subjects */
        EnforceAgent initialFailure =
                (EnforceAgent)agents.objs[random.nextInt(
                    numAgents)];
        /* the failure Bag will keep track of the
           superSovereigns to remove from
         * agents down the hierarchy whose immediate
           superSovereign doesn't fail */
        failures.add(initialFailure);
        failures.addAll(initialFailure.superSovereigns);

        /* all super sovereigns will lose this initialFailure's
           domain from
         * their own domains */
        Bag superSovereigns = initialFailure.superSovereigns;
        int numSupers = superSovereigns.numObjs;
        for(int superIndex=0; superIndex<numSupers; superIndex
            ++){
            ((EnforceAgent)superSovereigns.objs[superIndex]).
                domain
                    .removeAll(initialFailure.domain);
        }
        /* they lose their subjects and domain */
        initialFailure.numSubjects = 0;
        initialFailure.domain = new Bag();

        /* meanwhile, some sub-sovereigns lose everything and
           become freeMen
         * - I'll do this with a loop that constructs a Bag of
           each subsequent
```

```
 * tier, rather than by recursion that risks stack
    overflows (power law
 * of cascade sizes means some very big cascades, and
    deep recursions) */
Bag unenforced = initialFailure.subjects;
Bag lowerTierFail = new Bag();
Bag lowerTierSafe = new Bag();
while(unenforced.numObjs>0){
    EnforceAgent subject = (EnforceAgent)unenforced.
       objs[0];
    /* with a certain probability each subject will
       fail too */
    if(random.nextBoolean(contagionProb)){
        lowerTierFail.add(subject);
        failures.add(subject);
    }
    /* regardless of whether the subject fails, the
       enforcement
     * relationships from those above will be lost */
    else lowerTierSafe.add(subject);

    subject.superSovereigns = new Bag();

    unenforced.remove(subject);
}
while((lowerTierFail.numObjs>0) || (lowerTierSafe.
   numObjs>0)){
    Bag nextTierFail = new Bag();
    Bag nextTierSafe = new Bag();
    for(int subjectIndex=0;
            subjectIndex<lowerTierFail.numObjs;
            subjectIndex++){
        EnforceAgent sovereign
                = (EnforceAgent)lowerTierFail.objs[
                    subjectIndex];
        /* this sovereign loses their subjects and
           domain */
        sovereign.numSubjects = 0;
        sovereign.domain = new Bag();
        /* ...and each of their subjects will suffer
           the same fate */
        unenforced = sovereign.subjects;
        while(unenforced.numObjs>0){
            EnforceAgent subject = (EnforceAgent)
               unenforced.objs[0];
```

```
                /* with a given probability, this subject
                   fails too */
                if(random.nextBoolean(contagionProb)){
                    nextTierFail.add(subject);
                }
                else {
                    nextTierSafe.add(subject);
                }
                /* even if they do not fail, this subject
                   will still lose
                 * their relationships with the enforcers
                    above */
                subject.superSovereigns = new Bag();
                unenforced.remove(subject);
            }
        }
        for(int sovereignIndex=0;
                sovereignIndex<lowerTierSafe.numObjs;
                sovereignIndex++){
            EnforceAgent sovereign
                    = (EnforceAgent)lowerTierSafe.objs[
                        sovereignIndex];
            /* Each of their subjects will also lose any
               sovereigns who have
             * failed above them
             * (the Bag passed here will also contain
                sovereigns who have
             * failed in different branches of the
                enforcement hierarchy,
             * but as they cannot be enforcers in any other
                 way there is no
             * harm passing them to the removeAll method */
            Bag lessEnforced = sovereign.subjects;
            for(int subjectIndx=0;
                    subjectIndx<lessEnforced.numObjs;
                    subjectIndx++){
                EnforceAgent subject
                        = (EnforceAgent)lessEnforced.objs[
                            subjectIndx];
                /* this subject's subjects will also lose
                   superSovereigns */
                nextTierSafe.add(subject);
                /* and this subject must lose those
                   supersovereigns that
                 * have failed */
```

```
                    subject.superSovereigns.removeAll(failures)
                        ;
                }
            }
            lowerTierFail = nextTierFail;
            lowerTierSafe = nextTierSafe;
        }
    }

    /* the formal economy will presumably be the largest
       component, output will
     * therefore be proportional to the number of pairs of
        agents in that
     * component */
    public double getOutput(){
        double output = 0;
        /* refresh the record of components, following growth
           and collapse*/
        discoverComponents();

        /* move through the components one by one, calculating
           the output and
         * keeping track of the highest; this will be the
            formal economy */
        int numComponents = components.numObjs;
        for(int componentIndex=0; componentIndex<numComponents;
            componentIndex++){
            int componentSubjects = ((Bag)components.objs[
               componentIndex])
                    .numObjs - 1;
            double componentOutput
                    = (double)componentSubjects*(double)(
                        componentSubjects-1);
            if(componentOutput>output)
                output = componentOutput;
        }
        return output;
    }

    void discoverComponents(){
        /* refresh the bag of components */
        components = new Bag();

        /* any agent with no superSovereigns will be either
           their own component
```

```
         * or the sovereign to a component, so we need only
           explore the
         * components of such agents */
        for(int agentIndex=0; agentIndex<numAgents; agentIndex
           ++){
            EnforceAgent agent = (EnforceAgent)agents.objs[
               agentIndex];
            if(agent.superSovereigns.isEmpty())
            {
                Bag newComponent = new Bag(agent.domain);
                newComponent.add(agent);
                components.add(newComponent);
                int numMembers = newComponent.numObjs;
                for(int memberIndex=0; memberIndex<numMembers;
                   memberIndex++)
                {
                    ((EnforceAgent)newComponent.objs[
                       memberIndex]).component
                            = newComponent;
                }
            }
        }
    }

    int fastIntPow(int base, int exponent){
        int power = 1;
        for(int product=0; product<exponent; product++){
            power *= base;
        }
        return power;
    }
}
```

Chapter 4

The Power of Indirect Inference under Long Memory

4.1 Introduction

This chapter takes as its starting point a question in the conclusion of the previous chapter: does the method of Indirect Inference Testing have power to reject bad descriptions of data when it has significant autocorrelation between distant observations —a property known as *Long Memory*, explained in section 4.3.

As explained in section 4.2, Indirect Inference involves two different models, one reductionist model that relates the data to microfoundations, and one statistical *auxiliary model* that is used for comparison of the data and the first model. This allows long memory to play two very different roles: it can be a feature of the data and reductionist model, or it can be a feature of the auxiliary model used to describe them. I explore both these roles. In keeping with chapter 3's use, however, I restrict attention to univariate data, specifically economic output.

In order to make this investigation relevant to the wider Economics community, I choose reductionist models, with and without long memory, that are already popular in the literature —described in detail in section 4.4. I mirror this in choosing the most standard of auxiliary models, in the form of an Autoregressive process and a Fractionally Integrated Autoregressive process —described in section 4.3.

In section 4.5, the investigation finds that Indirect Inference Testing, using a long-memory auxiliary model, has reasonable power to distinguish good descriptions of long memory data. It also offers an improvement over a short memory auxiliary model for data without long memory. However, I find that for data without long memory Indirect Inference testing has very little power generally.

To my knowledge, this is the only investigation of the effects of long memory on Indirect Inference.

4.1.1 Notation

Throughout, let X_t denote a vector of observations from some process. Let L denote the *lag operator* such that $LX_t = X_{t-1}$ and $L^{-1}X_t = X_{t+1}$. ϵ_t will denote an independent normal innovation, $\sim N(0, \sigma^2)$.

4.2 Indirect Inference and Testing

Indirect Inference describes techniques wherein observed data and a structural model, based on theory, are described using the same auxiliary model and the descriptions are compared. The auxiliary model here would usually be a statistical description of data such as (joint) moments or their VAR representation, but could instead be any function of data that was deemed important, such as Macroeconomics' "stylised facts". It is most obviously useful, above conventional methods, where calculation or estimation of a likelihood function for the theoretical model is not possible, but cheap simulation is. This simulation allows samples from the structural model to be generated. The auxiliary model can be fitted to such samples and its parameters compared to those fitted to the observed data.

Gregory and Smith (1991) were the first to use Indirect Inference to estimate the parameters of a theoretical model. Explicitly, such estimation selects parameters for a the-

oretical model, that *best* reproduce observed phenomena, according to the following recipe:

1. The auxiliary model is fitted to the observed data;
2. Next, the auxiliary model is fitted to data from simulations of the structural model for various parameter values;
3. These structural data fittings are compared to the fitting for the observed data, and structural parameters are chosen that lead to auxiliary parameters as close as possible to those for the observed data, according to some measure.

Other examples can be found in Gourieroux et al. (1993) and others. This chapter does not make use of this method, and it is included here only for clarification and contrast with the actual method of interest, Indirect Inference Testing. Indirect Inference Testing is a more recent adaptation of this approach to allow the appraisal and comparison of different models' efficacy. This appraisal is based on the classical statistical method of hypothesis testing: it sets a bar that a theoretical model must clear in order to be considered a good description of the data. The intuition is that if a theoretical model correctly captures the relationship between the phenomenon and its components then it should have the same description as our observation of that phenomenon, when using the auxiliary model —for example, the same moments as the real-world observed data. The bar chosen in practice is a lower limit on the likelihood of auxiliary parameters more extreme than those observed: it checks that, were the theoretical model correct, there would be a high probability of seeing more extreme data than we observed. Using simulations of the theoretical model to sketch out a distribution for the auxiliary model's parameters, we then have the following recipe:

1. The auxiliary model is fitted to the observed data;
2. The structural model is simulated many times to get a distribution of samples from it;

3. The auxiliary model is fitted to each simulated sample to give a distribution for its parameters, θ, under the null hypothesis that this parameterisation of the structural model is *true*;

4. We count the number of simulated cases (the fraction of the distribution) for which the auxiliary parameters are more extreme than the observed auxiliary parameters; if this is fewer than a certain significance level then we reject this parameterisation of the theoretical model as *correct* —it is a bad description of the data.

The extremity of the auxiliary parameters' deviation cannot be measured directly, because setting separate confidence bounds for each parameter could not guarantee a given significance level without taking their correlations into account. That is, if we choose critical values in each of p parameters' marginal distributions so as to reject $5\%/p$ of true models, then we will only reject 5% of all true models if the parameters are uncorrelated. Rather than adjust the critical values to take account of correlations between variables, it is more convenient to combine all the parameter values into a single statistic that has a well behaved distribution. From the central limit theorem, it can be shown that a Wald statistic,

$$W = \theta' \Sigma^{-1} \theta$$

is distributed according to a χ^2 distribution, giving it smooth exponential tails that allow clear rejection of models that give a value too far into the upper tail. The concept of *truth* about future observations, central to hypothesis testing, is trickier than natural language would have us believe. As I explained in chapter 2, this is because we have no technology that can identify such truth, and hence no way of appraising guesses at it or methods of guessing. This means that a methodology whose objective is identifying truth cannot be justified by its own criterion — which fatally undermines positive science. Such a criticism of hypothesis testing is not fatal, because the method can be viewed in a different light: it can be seen as ensuring that an observed statistic can be

described in at most a given length of description. Chapter 2 explains that assumptions about the future based on a Minimum Description Length can be self-consistent because it efficiently describes existing predictive methods and thereby advocates their, and its own, continued use. I explored this methodological issue in more depth in chapter 2, along with its implications for Macroeconomics. But, for the purposes of this study it need not bear further thought.

Throughout this paper I remain close to the method employed in Le et al. (2012), for assessing the power of rival methods. Their comparison is between Indirect Inference Testing, Likelihood Ratio, and the Del Negro-Schorfheide DSGE-Var weight, and they find that Indirect Inference has comparatively high power. My study takes for granted the validity of Indirect Inference, and addresses a more specific question about the paradigm: I restrict my attention to different Auxiliary models within the Indirect Inference Testing method.

4.3 Long Memory

Long memory is defined in many ways across the disciplines in which it is used (Samorodnitsky, 2007), but the general sense is of a process for which observations long past still carry relevant information on future behaviour: statistical dependence decays only slowly with time separation. Some more specific definitions are:

1. slower-than-exponential decay of the autocorrelation function;

2. slower-than-exponential decay of the power spectral density;

3. extremal power spectral density close to zero.

Making those first two definitions even more explicit, much of the literature on long memory focusses on processes with autocorrelation and power spectral densities that

decay hyperbolically, because they become Gaussian noise once fractionally differenced (i.e. X_t s.t. $(1-L)^d X_t \sim N(\mu, \sigma^2)$ for $d \in (0,1)$). It is such a hyperbolic decay model that I will use as an exemplar of long memory dynamics in section 4.4. Meanwhile, it is the last, loosest, definition that will be exploited by the Local Whittle Estimator that I employ in section 4.5: focussing on the power spectral density only around zero gives it the power to describe models with hyperbolic long-memory in the tail, but remain agnostic about shorter term dynamics.

A proper description of long memory was first developed by Benoit Mandelbrot in explanation of price movements (Mandelbrot, 1969), and the observations of Hurst (1951) in hydrology. Hurst was studying the flow of water in the Nile through its *Rescaled Range Statistic*,

$$\frac{R}{S}(X_1, ..., X_n) = \frac{max_{0\leq i\leq n}(S_i - \frac{i}{n}S_n) - min_{0\leq i\leq n}(S_i - \frac{i}{n}S_n)}{\left(\frac{1}{n}\sum_{i=1}^{n}(X_i - \frac{1}{n}S_n)^2\right)^{\frac{1}{2}}}$$

For a process with exponentially decaying autocorrelation, the Central Limit Theorem dictates that his statistic should grow as the square root of the sample size. But, Hurst found that it consistently grew like $n^{0.74}$, which became an outstanding paradox known as the *Hurst Phenomenon* (Samorodnitsky, 2007). Meanwhile Benoit Mandelbrot had been examining financial time series; he found both that they were punctuated by dramatic changes, and that these changes were clustered together in time (Mandelbrot, 1983). These properties were profoundly non-Gaussian, and led Mandelbrot to a mathematics that would describe both phenomena characterised by hyperbolic rather than exponential distributions.

Despite arriving in economics before any other discipline, economists were very slow to address long memory with theory and it still goes unconsidered by much of the orthodoxy. Physics on the other hand did not have a formal description for long memory until Mandelbrot later imported it, but then began to identify processes that could generate it like Self Organised Criticality (Bak et al., 1987). Whether models are based on Self-Organising Criticality, a Shot noise process (Lowen and Teich, 2005), a cluster-

ing Poisson point process (Grüneis and Musha, 1986), collections of reversible Markov Chains (Erland and Greenwood, 2007), or its special case of combined AutoRegressive processes (Granger, 1980), they generally rely on the aggregation of many components. Indeed, it is aggregation which has so far been used in Economic models to produce long memory.

Bak et al. (1993) import the mechanism of self-organised criticality into economics, in a model that aggregates the discontinuous inventory dynamics of the many intermediate producers in an economy, and finds *avalanches* of production activity that give rise to long memory. In order to reconcile unit root findings with beta convergence, Michelacci and Zaffaroni (2000) develop a heterogeneous sector Solow-Swan growth model that displays long memory on aggregation. A long memory Autocorrelation function very much like that of economic data is produced by Abadir and Talmain (2002), through the non-linear aggregation of monopolistically competitive firms — that is, the arithmetic mean is taken over productivities that are autoregressive in logs, and so grow geometrically. Davidson and Sibbertsen (2005) show that long memory could arise from the aggregation of firms represented by short memory processes that have means that change randomly at power law distributed intervals. Perhaps the most approachable model thus far sees Haubrich and Lo (2001) make use of the result of Granger (1980) to derive long range dependence from firms' random linear production technology, distributed according to a Beta distribution. It is this last model that I will use as an exemplar of theoretical long memory models, for the appraisal of Indirect Inference Testing in section 4.4.

4.3.1 Fractional Integration

The canonical long memory process is, as mentioned above, one with Gaussian Fractional differences, $I(d)$ with $d \in (0, 1)$, known as Fractional Integration (Granger and

Joyeux, 1980):

$$(1-L)^d X_t = u_t \mathbb{1}(t>0) \text{ where } u_t \sim N(\mu, \sigma^2) \text{ and } t \in \mathbb{N}$$

$$X_t - dX_{t-1} + \frac{d(d-1)}{2!}X_{t-2} - ... = u_t$$

Because the differencing parameter, d, lies in the unit interval, these models form a neat bridge between the more familiar econometric concepts of unit root ($d = 1$) and canonical time series ($d = 0$). But, this bridge does not display a neat spectrum of behaviour: there is an important discontinuous transition at $d = 0.5$ between stationary processes for $d < 0.5$, and for $d > 0.5$ those which are non-stationary (like unit root processes) but still mean reverting (like a canonical short memory time series). That is, as d becomes greater than 0.5 departures from mean can suddenly be so prolonged that the variance becomes infinite — because the probability of data a given distance from mean decays no faster than the square of that distance grows. It is to this non-stationary category that macroeconomic output belongs, according to recent evidence (Mayoral, 2006). Such non-stationarity after the transition is not without complications for empirical investigation: most statistical techniques rely on the existence of a second moment. For this reason the estimators explored below evolve towards techniques that remove the effects of non-stationarity.

In the interval $d \in (0, 0.5)$ for which Fractional Integration is stationary, it can be shown that the autocorrelation function, $\rho(s)$, decays hyperbolically in the tail (Granger and Joyeux, 1980):

$$\rho(s) \sim s^{2d-1} \text{ for } s \to \infty, d \in (0, 0.5)$$

What's more, if we use $f(w)$ to denote the Spectral Density of a fractionally integrated process, it can be shown that this too follows a power-law for low frequencies (Granger and Joyeux, 1980):

$$f(\omega) \sim \kappa^{-2d} \text{ as } \omega \to 0$$

As an estimate of the Spectral Density of a process, the Periodogram for data drawn from a fractionally integrated model should therefore also be hyperbolic. These last two features are critical to the estimation procedures described below.

The famous critique by Lucas (1976), of the then prevailing macroeconometric paradigm, was based on the apparent instability of relationships in a vector difference equation of economic aggregates. This instability is precisely what would be expected in a short memory representation of a non-stationary process. Notably, however, modern Dynamic Stochastic General Equilibrium analysis doesn't address these time series properties either, because the model is generally reduced to a short memory VAR. This is yet another reason that such processes deserve more attention in the Economics literature.

4.3.2 AutoRegressive Fractionally Integrated Moving Average

Fractional integration provides the slower than exponential decay in autocorrelation demanded of a description of long memory, but it has very specific short run dynamics. In order to describe richer short term behaviour, the fractional integration can be combined with autoregressive and moving average behaviour, in much the same way that they combine with a random walk in an Error Correction Model:

$$X_t \sim ARFIMA(p,d,q) \qquad \Rightarrow \qquad \Phi(L)(1-L)^d X_t = \Theta(L)\epsilon_t$$

where,

$$\Phi(L) = 1 - \phi_1 L - ... - \phi_p L^p = 1 - \sum_{i=1}^{p} \phi_i L^i$$

and,

$$\Theta(L) = 1 + \theta_1 L + ... + \theta_q L^q = 1 + \sum_{i=1}^{q} \theta_i L^i$$

When in such a form, a difference operator allows the data to be differenced down to a strictly stationary $ARMA(p,q)$ — as opposed to one where the difference operator

was applied after the autoregressive polynomial. That is,

$$\Phi(L)(1-L)^d X_t = \Theta(L)\epsilon_t \qquad \Rightarrow \qquad \Phi(L)U_t = \Theta(L)\epsilon_t \text{ with } \mathbb{E}(U_t^2) < \infty$$

It therefore allows the use of conventional estimation techniques for the parameters of Φ and Θ, once the difference parameter, d, has been estimated.

4.3.3 Estimators for Fractional Integration

Since long memory became a topic of interest several fundamentally different approaches have been followed, for the estimation of the fractional difference parameter, d. The most natural route to estimating the factional difference parameter would be to mimic the seminal observation of Hurst (1951) in hydrology or the subsequent analysis by Mandelbrot (1969). One approach that this would recommend is to estimate Hurst's Rescaled Range statistic. However, Bhattacharya et al. (1983) show that this estimator cannot cope with non-stationary series. A second approach from simple statistical observations on long-memory series is that of Aggregated Variances: a fractionally integrated series is divided into bins of equal length, such that their means themselves represent a series; the variance of these means then has a power-law relationship to their size if the series was fractionally integrated; this relationship makes the ratio of log-variance to log-bin size an estimator of the difference parameter. This method fails more dramatically, as Giraitis et al. (1999) show it to be systematically biased.

Perhaps an equally natural approach is that of Geweke and Porter-Hudak (1983). This makes use of the fact that a power law distribution, like the periodogram for a fractionally integrated process, is just a straight line with a slope of $2d - 1$ on a log-log plot. Thus, taking logs of both frequency components and a deterministic regressor, one can use linear regression to estimate the differencing parameter. An obvious drawback would be that on a log-log plot those few observations in the heavy tail of a power-law distribution have undue influence on the estimate of the slope, leading to systematic

bias (Clauset et al., 2009). Geweke and Porter-Hudak (1983) counter this by only regressing on the lowest frequency components. A drawback that is not overcome is the inefficiency of this method (Velasco, 1999), although Giraitis et al. (1999) prove its asymptotic normality and unbiasedness.

An estimator that is based on the likelihood of a fractionally integrated process is developed by Sowell (1992). A more tangible benefit of this approach is that AR and MA parameters of an ARFIMA model can be estimated at the same time as the differencing parameter, avoiding bias from estimating each separately. Despite the obvious gains from working with an exact likelihood function, the procedure is dependent on the existence of second moments, and therefore invalid for non-stationary processes.

An alternative, computationally more efficient, likelihood approach is based around the Whittle approximation of the likelihood function for a Gaussian series — like the correctly differenced version of the data, $(U_t)_{t>0}$, according to the Wold Decomposition. Further, it is semi-parametric, allowing it to estimate the long memory properties independently of short run dynamics. The likelihood, for parameters ζ (including the fractional difference parameter), n-dimensional differenced data, U, and covariance matrix, $\Sigma_{n,\zeta}$, is well known to be,

$$L(\zeta) = (2\pi)^{-n/2} \frac{1}{\sqrt{|\Sigma_{n,\zeta}|}} \exp\left(-\frac{1}{2} U x' \Sigma_{n,\zeta^{-1}} U x\right)$$

The evaluation here of the inverse of the covariance matrix is computationally expensive, and may be prohibitive for large data sets. Instead, we can recognise that all the left-right diagonals of the covariance matrix are constant, because they are assumed to follow the same marginal process. This makes the covariance matrix, $\Sigma_{n,\zeta}$ a *Toeplitz* matrix, which in turn means that its eigenvectors will be well approximated by the basis of a discrete fourier transform of the differenced data, and its eigenvalues by the corresponding spectral density, f_U, components and periodogram, I_U. Inverting the diagonalised covariance matrix then produces an expression in the spectral density

values that is the Whittle Approximation to the likelihood,

$$\log L(\zeta) \approx 2n \log 2\pi + \sum_{j=0}^{n-1} \left(\log f_U(\omega_j, \zeta) + \frac{I_U(\omega_j)}{f_U j(\omega_j, \zeta)} \right)$$

Where $\omega_j = 2\pi j/n$. Clearly we don't have the periodogram for U_t, only for X_t. This is fixed by replacing I_U with an approximation, namely $\omega_j^{2d} I_X$, and the Jacobian term $\sum_{j=1}^{m} \log \omega_j^{-2d}$ (Shimotsu, 2010). As only the fractional difference parameter, d in ζ, will be relevant at very low frequencies, ω, restricting attention to these frequencies allows us to ignore short run dynamics while maximising the likelihood contribution of d. This then involves minimising the negative term in the likelihood (Velasco, 1999),

$$R_m(d) = \log \left(\frac{1}{m} \sum_{j=1}^{m} \omega_j^{2d} I(\omega_j) \right) - 2d \frac{1}{m} \sum_{j=1}^{m} \log \omega_j$$

Fox and Taqqu prove that the Whittle estimator is consistent and asymptotically normal, while Dahlhaus (1989) proves its efficiency.

The Whittle Estimator is still not robust to non-stationarity when $d > 0.75$, but two workarounds have been developed. The first is to treat the data with a *taper* beforehand, so as to remove non-stationarity in a controlled way (Hurvich and Ray, 1995). A taper is a sequence of weights that are multiplied by the elements of a time series before Fourier transformation, usually in order to reduce the effects of outliers on the spectral density estimates. In this context, however, the taper reduces the departures from mean of a non-stationary or polynomial-deterministic process in a controlled way. This way the difference parameter's relationship with the spectral density is unchanged, and the Whittle Estimator remains consistent (Velasco, 1999). An unfortunate implication of the tapering is that the variance of the estimator is significantly greater (Hurvich and Chen, 2000).

The second workaround for non-stationarity in the Whittle estimator is to replace an estimate of the periodogram in the conventional Whittle Approximation with an Exact

expression (Shimotsu, 2010). This leads to the Exact Whittle Estimator based on the minimisation of the following objective function, for $\Delta^d X_t = (1-L)^d X_t$,

$$\hat{Q}_n(G,d) = \frac{1}{n}\sum_{j=1}^{n}\left(ln(f_{\Delta^d X}(\omega_j)\omega_j^{-2d}) + \frac{I_{\Delta^d X}(\omega_j)}{f_{\Delta^d X}(\omega_j)}\right)$$

This estimator is robust to non-stationarity and is efficient, unlike the Tapered Whittle. But, it relies on a known mean and the absence of a deterministic trend (Shimotsu, 2010).

To take advantage of the strengths of both the Tapered and Exact Whittle Estimators, they can be combined into a two-step procedure:

Step 1 - the Velasco (1999) Tapered Whittle is used to derive an initial estimate of the difference parameter; this estimate can then be used to partially-difference the data, and an estimate of the mean can then be selected from between the time series's mean (for $d < \frac{3}{4}$) and its first observation (for $d > \frac{1}{2}$), along with an estimate of any polynomial trend by regression on successive powers of a deterministic term;

Step 2 - next, having adjusted the data for the mean and polynomial trend, the Exact Local Whittle Estimator is used to derive an efficient estimate of the difference parameter, d.

Shimotsu (2010) shows that this estimator inherits the best of both individual Whittle estimators, with an efficient $N(0, \frac{1}{4})$ asymptotic distribution, and an unprecedented range of validity for underlying long memory, $d \in (-\frac{1}{2}, \frac{7}{4})$, even in the presence of an underlying polynomial time-trend.

4.4 Theoretical Models

In this paper I gauge the implications of long memory for Indirect Inference Testing, in both the auxiliary and theoretical models.

4.4.1 Long memory model

Because of its simplicity, the model of Haubrich and Lo (2001) is cited in several empirical works as an example of theoretical justification for interest in long memory econometrics (Sowell, 1992; Mayoral, 2006). The model builds on a conventional Real Business Cycle model, but produces long memory by using the aggregation result of Granger (1980) over heterogeneous sectors, each contributing a component to the $N \times 1$ output vector, Y_t, and consumption vector, C_t. The model is populated by N infinitely lived agents, each maximising the same quadratic personal utility function parametrised by a (diagonal) matrix B,

$$\sum_{t=0}^{\infty} \beta^t \left(C_t \prime \mathbf{1} - \frac{1}{2} C_t \prime B C_t \right)$$

The maximisation is limited by a resource constraint imposed by the economy's output,

$$Y_t = C_t + S_t \mathbf{1}$$

Where S is a matrix with $i, j^t h$ entry corresponding to the investment of sector i's product in sector j's production the next period. That production takes place based on the input-output productivity matrix A in a $VAR(1)$ process,

$$Y_t = A S_t \mathbf{1} + \eta_t$$

with $\eta_i \sim N(0, \sigma^2)$ $i \in [1, N]$. For convenience, it is assumed that each sector uses only its own output productively to produce more, making A diagonal. An analytical

solution to the agents' maximisation problem is then,

$$C_{it} = \frac{2\beta P_i A_{ii}^2}{2\beta P_i A_{ii}^2 - B_{ii}^2} Y_{it} - \frac{\beta q_i A_{ii} - 1}{B_{ii} - 2\beta P_i A_{ii}^2}$$

$$S_{it} = \frac{B_{ii}}{B_{ii} - 2\beta P_i A_{ii}^2} Y_{it} + \frac{\beta q_i A_{ii} - 1}{B_{ii} - 2\beta P_i A_{ii}^2}$$

with

$$P_i = B_{ii} \left(\frac{A_{ii} - \sqrt{(1+4\beta)A_{ii}^2 - 4}}{4\beta A_{ii}} \right)$$

and the q_i are constants. These solutions then give rise to the dynamics of output:

$$Y_{it+1} = \frac{A_{ii} B_{ii}}{B_{ii} - 2\beta P_i A_{ii}^2} Y_{it} + K_i + \eta_{it}$$

for constant K_i. Aggregation of these sectors gives rise to an ARMA(N,$N-1$) process for the aggregate. Clearly such a process is unmanageable for more than a few sectors. But if the square of the AR coefficient for Y_t follows a Beta(p,q) distribution, the aggregate parameters happen to be well approximated by those in the expansion of a fractionally integrated process,

$$X_t - dX_{t-1} + \frac{d(d-1)}{2!} X_{t-2} - ... = u_t$$

with $d \approx 1 - \frac{q}{2}$. Although this approximation will be good towards the limit, there is no reason to believe that the other aggregate parameter, p, will not affect autocorrelations for finite samples and at high frequencies. The theoretical model is fully identified, so given enough data its structural parameters could be estimated. For the sake of computational ease, I instead use Indirect Inference to test values for the aggregate parameters p, q, and σ.

4.4.2 Short memory model

Le et al. (2012) observe that further autoregressive terms improve the power of indirect inference testing for the popular short memory DSGE model of Smets and Wouters. Even though the DSGE model is reduced to a VAR, as is generally done for DSGE models, it should be no surprise that a higher order auxiliary VAR captures more model behaviour: the authors' investigation is limited to a subset of variables, and so their lower-dimensional VAR may therefore exclude sources of persistence from the larger reduced form VAR that are regained by increasing the order. It is therefore possible that the greater persistence of an ARFIMA auxiliary model may also better describe the model, and lend greater power to indirect inference in general. It is true that the Local Whittle estimator specifically excludes high frequency behaviour, but this is precisely why it could conceivably pick up contributions of more persistent but less immediately dominant variables omitted from the auxiliary VAR.

As I am following Le et al. (2012), it is appropriate that I base my assessment of auxiliary models on the same Smets-Wouters model. But, it is anyway an exemplar DSGE model, and one actually used in policy, so there could be no more worthwhile case to explore.

Since the prototypical Real Business Cycle model of Kydland and Prescott (1982), the DSGE class of macroeconomic models has grown to predominance within academia. These models seek to find the origin of whatever patterns may exist in and between aggregate variables, based ultimately on the principles of microeconomics. The main argument for such a method is the Lucas Critique: patterns in aggregate behaviour cannot be trusted to persist through changes in policy unless they are described in terms of fundamentals that cannot change — that is, the nature of human decision making (Lucas, 1976). What's more, a basis in microeconomics allows the microeconomic concept of welfare, as the satisfaction of agents' preferences by the allocation of scarce resources, to be imported into macroeconomics for the welfare assessment of policy changes. These models are subject to serious criticisms, which I gave in chapter 2,

but their currency (despite an abstraction from currency!) makes it important that an assessment of Indirect Inference not neglect them.

In all forms, the fluctuations of variables in DSGE models are driven by external economy-wide disturbances (*shocks*) to some of those variables — on top of steadier exogenous or endogenous gains to productivity driving up the stock variables. The RBC literature narrows these shocks to only those directly affecting productivity and preferences. Like most DSGE analyses, the RBC literature models the aggregate economy as if it behaved like a single rational decision maker, with perfect knowledge of the economy's physical constraints. On its own this of course means that historical values of flow variables don't contain information on the future. This feature is contradicted by the substantial serial correlation observed in their observed time series. Models therefore introduce *time to build*, delays in the realisation of stock variable adjustments, and *capital utilisation rates*, leaving capital stock variables in place but unemployed.

Further dependence on past states is introduced in the more elaborate DSGE models of the New Keynesian school with the microfoundations expanded to allow excess demand or supply, as proposed in Hart (1982). This then leads to agents on one side of the market facing a maximum constraint on their quantity, and market output being equal to the minimum of these:

$$\text{Quanitity} = \min(\text{Supply}(\text{Price}), \text{Demand}(\text{Price}))$$

This is brought into a dynamic setting by the *staggering* of Taylor (1979), made more tractable by Calvo (1983): having the prices update to an equilibrium value for only a (stochastic) fraction of firms over a given period, so that the others face a price that leaves excess supply or demand.

The Smets-Wouters model is described by the authors as an extension of the small scale models above, and indeed it includes all those features mentioned. After the optimisa-

tion problem has been reduced to Euler conditions, and these log-linearised around a long run steady-state, the model consists of several simple linear equations that can be represented by a structural VAR. For output, y_t, consumption, c_t, investment, i_t, and capital (k_t) utilisation rate, z_t, with steady state values denoted by a subscript, x_y, and unanticipated change, ϵ_t^g, to government spending, g, we have the aggregate resource constraint,

$$y_t = c_y c_t + i_y i_t + z_y z_t + \epsilon_t^g$$

$$c_y = 1 - g_y - i_y$$

$$i_y = (\gamma - 1 + \delta)k_y$$

$$z_y = R_\star^k k_y$$

$$\epsilon_t^g = \rho_g \epsilon_{t-1}^g + \eta_t^g + \rho_g a \eta_t^a \quad \eta_t^x \sim N(0, \sigma_x^2)$$

Where $R_\star^k$ is the steady state rental rate for capital, γ is the external productivity growth rate for the economy, and everything else is a parameter to be estimated. The first two equations are accounting identities, while the third tells us that investment compensates for depreciation, and then goes on to take advantage of the productivity growth. The auto behaviour of the To maximise a non-separable utility function in leisure, l_t, and goods consumption (relative to habit), the representative agent follows consumption plan,

$$c_t = c_1 c_{t-1} + (1 - c_1)\mathbb{E}_t c_{t+1} + c_2(l_t - \mathbb{E}_t l_{t+1}) - c_3(r_t - \mathbb{E}_t \pi_{t+1} + \epsilon_t^b)$$

$$c_1 = \frac{\lambda/\gamma}{1 - \lambda/\gamma}$$

$$c_2 = \frac{(\sigma_c - 1)W_\star^h L_\star / C_\star}{\sigma_c(1 + \lambda/\gamma)}$$

$$c_3 = \frac{1 - \lambda/\gamma}{\sigma_c(1 + \lambda/\gamma)}$$

$$\epsilon_t^b = \rho_b \epsilon_{t-1}^b + \eta_t^b$$

Where λ describes the degree to which habit affects preferences, and σ_c parametrises the relationship between consumption and leisure in the utility function. ϵ_t^b is the agent's error in their forecast of the real interest rate, r_t. While the first two terms clearly correspond to habit and to anticipated future consumption, the third is not so transparent; it shows the dependence of consumption on the anticipated growth in hours worked for the next period. The investment decision involves the real value of the capital stock, q_t, and two new parameters: the rate at which the consumer exponentially discounts the present value of future consumption in their plans, β; and ϕ, the steady state elasticity of the capital adjustment cost function.

$$i_t = i_1 i_{t-1} + (1 - i_1)\mathbb{E}_t i_{t+1} + i_2 q_t + \epsilon_t^i$$

$$i_1 = \frac{1}{1 - \beta\gamma^{1-\sigma_c}}$$

$$\frac{1}{(1 + \beta\gamma^{1-\sigma_c})\gamma^2\phi}$$

That real value of capital is determined by a no-arbitrage argument, captured in the linear equation,

$$q_t = q_1 \mathbb{E}_t q_{t+1} + (1 - q_t)\mathbb{E}_t r_{t+1}^k - (r_t - \pi_{t+1} + \epsilon_t^b)$$

$$q_1 = \beta\gamma^{-\sigma_c}(1 - \delta)$$

For a real rental rate for capital of,

$$r_t^k = -(k_t - l_t) + w_t$$

Output in the economy is produced using capital and labour (l_t), according to,

$$y_t = \Phi_p(\alpha k_t^s + (1 - \alpha)l_t + \epsilon_t^a)$$

Where productivity, ϵ_t^a is an autoregressive process, $\epsilon_t^a = \rho_a \epsilon_{t-1}^a + \eta_t^a$. The *time to build*

feature described above, means that capital doesn't contribute to output for one whole period after its purchase. Combined with the partial utilisation of capital, this is why the equation features effective capital instead:

$$k_t^s = k_{t-1} + z_t$$

That capital utilisation rate takes on a simple form when minimised as $z_t = z_1 r_t^k$, with z_1 a function of the elasticity of capital utilisation adjustment cost. The base capital doesn't accumulate simply either, with an unpredictable efficiency of investment compounding the effects of depreciation:

$$k_t = k_1 k_{t-1} + (1 - k_1) i_t + k_2 \epsilon_t^i$$

where $k_1 = (1 - \delta)/\gamma$ and $k_2 = (1 - (1 - \delta)/\gamma)(1 + \beta\gamma^{1-\sigma_c})\gamma^2\phi$. The New Keynesian feature of delayed price adjustment gives rise to the *Phillips Curve* relationship between inflation, π_t, and output (via the capital-labour ratio, productivity shock, and real wage, w_t):

$$\pi_t = \pi_1 \pi_{t-1} + \pi_2 \mathbb{E}_t \pi_{t+1} - \pi_3 [\alpha(k_t^s - l_t) + \epsilon_t^a - w_t] + \epsilon_t^p$$

with $\pi_1 = \frac{\iota_p}{1+\beta\gamma^{1-\sigma_c}\iota_p}$, $\pi_2 = \frac{\beta\gamma^{1-\sigma_c}}{1+\beta\gamma^{1-\sigma_c}\iota_p}$, and

$$\pi_3 = \frac{1}{1 + \beta\gamma^{1-\sigma_c}\iota_p} \frac{(1 - \beta\gamma^{1-\sigma_c}\xi_p)(1 - \xi_p)}{\xi_p(1 + (\phi_p - 1)\epsilon_p)}$$

The Smets Wouters set up doesn't strictly follow the New Keynesian staggering, as firms that do not revise their prices to the equilibrium still update their prices according to simple rule based on past inflation; ι_t refers to the degree to which firms index on past inflation. The mean rate of price revision is itself captured by the ξ_p parameter, while ϵ_p describes the degree of disparity among the production scales of different industries. The same New Keynesian features are also present in the labour market,

giving rise to an analogous wage rate inflation:

$$w_t = w_1 w_{t-1}(1-w_1)(\mathbb{E}_t w_{t+1}+\mathbb{E}_t \pi_{t+1}) - w_2 \pi_t w_3 \pi_{t-1} - w_4 \left(w_t - \left(\sigma_l l_t + \frac{1}{1-\lambda}(c_t - \lambda c_{t-1}) \right) \right) + \epsilon_t^w$$

Where $w_1 = \frac{1}{1+\beta\gamma^{1-\sigma_c}}$, $w_2 = \frac{1+\beta\gamma^{1-\sigma_c}\iota_w}{1+\beta\gamma^{1-\sigma_c}}$, $w_3 = \frac{\iota_w}{1+\beta\gamma^{1-\sigma_c}}$, and

$$w_4 = \frac{1}{1+\beta\gamma^{1-\sigma_c}} \frac{1-\beta\gamma^{1-\sigma_c}\xi_w}{\xi_w(1+(\phi_w-1)\epsilon_w)}$$

Again, the presence of lagged wages is due to suppliers who can't equilibrate their prices following a simple rule instead. Inflation and expected inflation have obvious roles, while the second to last term represents the average margin added to prices by the monopolistic but partially-substitutable firms. For both inflation and wages, the unpredictable change ϵ_t^x follows an ARMA(1,1) process, to atheoretically improve the description of data. The final equation in the model represents the actions of the monetary authority in setting the interest rate, in response to inflation, output, and the *potential output*, y_t^p, that would eventually prevail were free entry to markets allowed to undermine the market power of monopolists without being disturbed by the unforeseen disturbances ϵ_t^w and ϵ_t^p:

$$r_t = \rho r_{t-1}(1-\rho)\left(r_\pi \pi_t + r_y(y_t - y_t^p)\right) + r_{\Delta y}\left((y_t - y_t^p) - (y_{t-1} - y_{t-1}^p)\right) + \epsilon_t^r$$

Such a *Taylor Rule* is intended to represent Monetarist interventions in the economy, with ϵ_t^r being the unpredictable errors in this intervention.

4.5 Power of Indirect Inference Testing

In order to avoid the complications of fractional cointegration (Aloy and de Truchis, 2012; Gil-Alana, 2003), and to stay close to my previous use of Indirect Inference with long memory in chapter 3, I restrict attention to testing models through the behaviour

of a single variable, economic output. Java codes for the simulation of models are available in the online appendix. Java codes for the implementation of Indirect Inference are also available, and each stage of this statistical processing was cross-checked using other applications.

4.5.1 Short versus Long memory Auxiliary Models for a Long Memory Theoretical Model

In order to assess the power of Indirect Inference for a parameterisation of the model that is relevant to practice, I choose parameter values that are likely to produce an autocorrelation function exponent in the order of US GNP. Because long memory relates observations over very long lags, it is important to use the longest data sets available. For this reason I employ the, efficient, Two-Step Exact Local Whittle estimator on the historical US GDP series of Maddison, which spans 138 years. The fractional difference parameter is estimated as *0.85*. With this estimate in hand, Haubrich and Lo (2001) suggests beta parameters of around 0.2, which I pair with a unit shock variance for simplicity. To assess the test's power, I test a false model with parameters that deviate by a range of percentages including both trivial adjustments, and the extreme case of stationary behaviour ($d = 0.4$). These deviations are positive, because the restriction of the Beta distribution's parameters to positive values prevents the Haubrich-Lo model from having qualitatively different behaviour with reductions in its parameters.

The results of the Monte Carlo experiments, in table 4.1 are broadly as one might expect, given the implied deviations in difference parameter shown in the second column. Despite my cautious speculation in the conclusion of chapter 3, indirect inference testing using long memory auxiliary models has some power to identify misspecification of a long memory theoretical model. That is, the test always abandons models when the parameterisation is so different that it enters another phase, with different statistical properties. Here, the model has stationary behaviour for $d < 0.5$, and borderline behaviour at $d = 0.5$. Meanwhile the rejection rate is reasonable for misspecifications

		Rejection Rates			
%deviation	$\Longrightarrow d$	$AR(1)$	$ARFIMA(1,d,0)$	$AR(3)$	$ARFIMA(3,d,0)$
0	0.90	0.05	0.05	0.05	0.05
10	0.89	0.04	0.07	0.05	0.08
20	0.88	0.03	0.10	0.05	0.10
100	0.80	0.01	0.54	0.08	0.52
200	0.70	0.58	0.93	0.15	0.92
300	0.60	0.94	0.99	0.27	0.99
400	0.50	1.00	1.00	0.50	1.00
500	0.40	1.00	1.00	1.00	1.00

Table 4.1: Rejection rates for long memory Data Generating Process, using different auxiliary models.

that would throw out the difference parameter by more than 10% — that is, parameter values more than 100% greater than the realistic ones chosen.

For short memory auxiliary models, the story is quite different. Decisively different behaviour, where $d < 0.5$ can be firmly rejected, but there is little power otherwise, even at the borderline case of $d = 0.5$ where non-stationarity is lost. Again, this is what might have been expected given the quite different nature of short and long memory processes. What is unexpected is that more AutoRegressive terms can actually reduce the power of Indirect Inference to spot misspecification; comparison of the third and fifth columns shows that for significant parameter changes the $AR(3)$ actually has less power than the $AR(1)$. The reason for this effect is unclear, but it is seen to a minor extent in its effects on the long memory auxiliary tests, where the addition of more short memory structure leads to a (minor) reduction in power. That is, more AutoRegressive terms can actually impede the performance of long memory auxiliary models for substantial misspecification that does not change the qualitative behaviour of the theoretical model.

4.5.2 For a Short Memory Theoretical Model

Again, I choose a parameterisation that would be relevant to practice. In this case, I follow Le et al. (2012) in using that from the original Smets and Wouters (2007). Likewise, the range of percentage deviations for the false model's parameters is chosen to parallel that in Le et al. Their results showed that testing three variables with a VAR auxiliary model gives reasonable power, with rejection of 100% of false models when misspecification was more than 10%.

Consistent with the findings of Le et al. (2012), I find that additional autocorrelation structure in the auxiliary model generally improves the power of indirect inference testing on the Smets Wouters model - i.e. its ability to reject incorrect parameterisations. This is the case regardless of whether the additional autocorrelation is described using short or long memory auxiliary models. That fractional differencing would do this was not a foregone conclusion, because the Smets Wouters model reduces to a VAR with exponential decay in its autocorrelation function that is very different from the hyperbolic decay of a fractionally differenced process. Nevertheless, it is clear that the improvement in approximating short term autocorrelation structure is greater than any loss of accuracy from misspecifying longer term persistence — at least for the lengths of time series for which we actually have macroeconomic data. Table 4.2 shows the power to reject models with parameters perturbed by various percentages when using four different auxiliary models: an $AR(1)$, $ARFIMA(1, d, 0)$, $AR(3)$, and $ARFIMA(3, d, 0)$. By comparing the $AR(p)$ and $ARFIMA(p, d, 0)$ columns, we see the clear gains from introducing the difference parameter over a short memory autoregressive auxiliary model.

By comparing the second and third columns of table 4.2, we see that the introduction of the difference parameter actually improves power by more than the introduction of two more autoregressive terms, with the false model often rejected for substantially more simulations. This suggests that the autoregressive expansion of the long memory

	Rejection Rates			
%deviation	$AR(1)$	$ARFIMA(1,d,0)$	$AR(3)$	$ARFIMA(3,d,0)$
0	0.05	0.05	0.05	0.05
1	0.13	0.71	0.28	0.71
3	0.17	0.58	0.32	0.58
5	0.50	0.69	0.23	0.69
7	0.01	0.28	0.16	0.28
10	0.58	0.74	0.17	0.74
15	0.21	0.70	0.13	0.70
20	0.17	0.27	0.42	0.27

Table 4.2: Rejection rates for short memory Data Generating Process using different auxiliary models.

auxiliary model,

$$y_t = \epsilon_t + dy_{t-1} + \frac{d(d-1)}{2!}y_{t-2} + ...$$

improves the approximation of the autoregressive expansion for a single variable from the 10-variable Smets-Wouters reduced form $VAR(1)$. Or at least, it improves relative to just a few autoregressive terms in a small AR model, even if this remains a poor approximation. To illustrate how this could be possible, figure 4.1 shows the autoregressive representation of a $VAR(1)$ alongside the autoregressive representations for each of the auxiliary models ($AR(3)$ and $ARFIMA(1,d,0)$). The parameterisations were chosen to produce similar behaviour, but clearly the AR(3) can not reproduce the slow decay of autoregressive terms effectively seen for each variable in a Vector AutoRegression. In Indirect Inference the role of the auxiliary model is to describe the data generally, not necessarily in the most efficient way that identifies the most regularities, as would be the case if we were using the auxiliary model for induction. For this reason, the relative values of the two auxiliary models is not as simple as the lower reduction in the degrees of freedom that come with one difference parameter over two autoregressive parameters.

Despite the performance gained by using a long memory auxiliary model, it is important to note that in all cases indirect inference has very low power to reject false hypotheses on a single variable. We see in table 4.2 that even as parameters are perturbed by

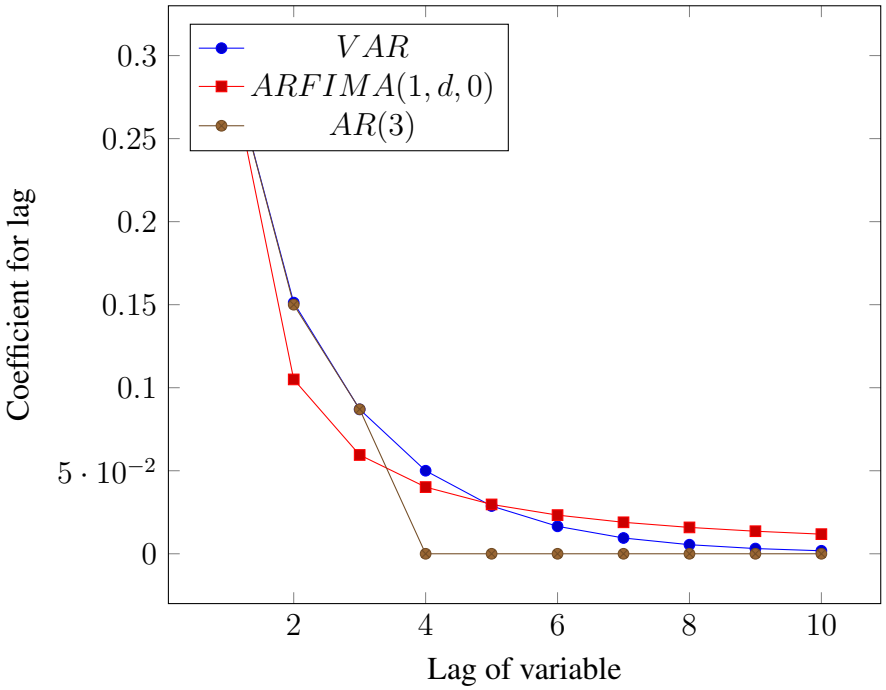
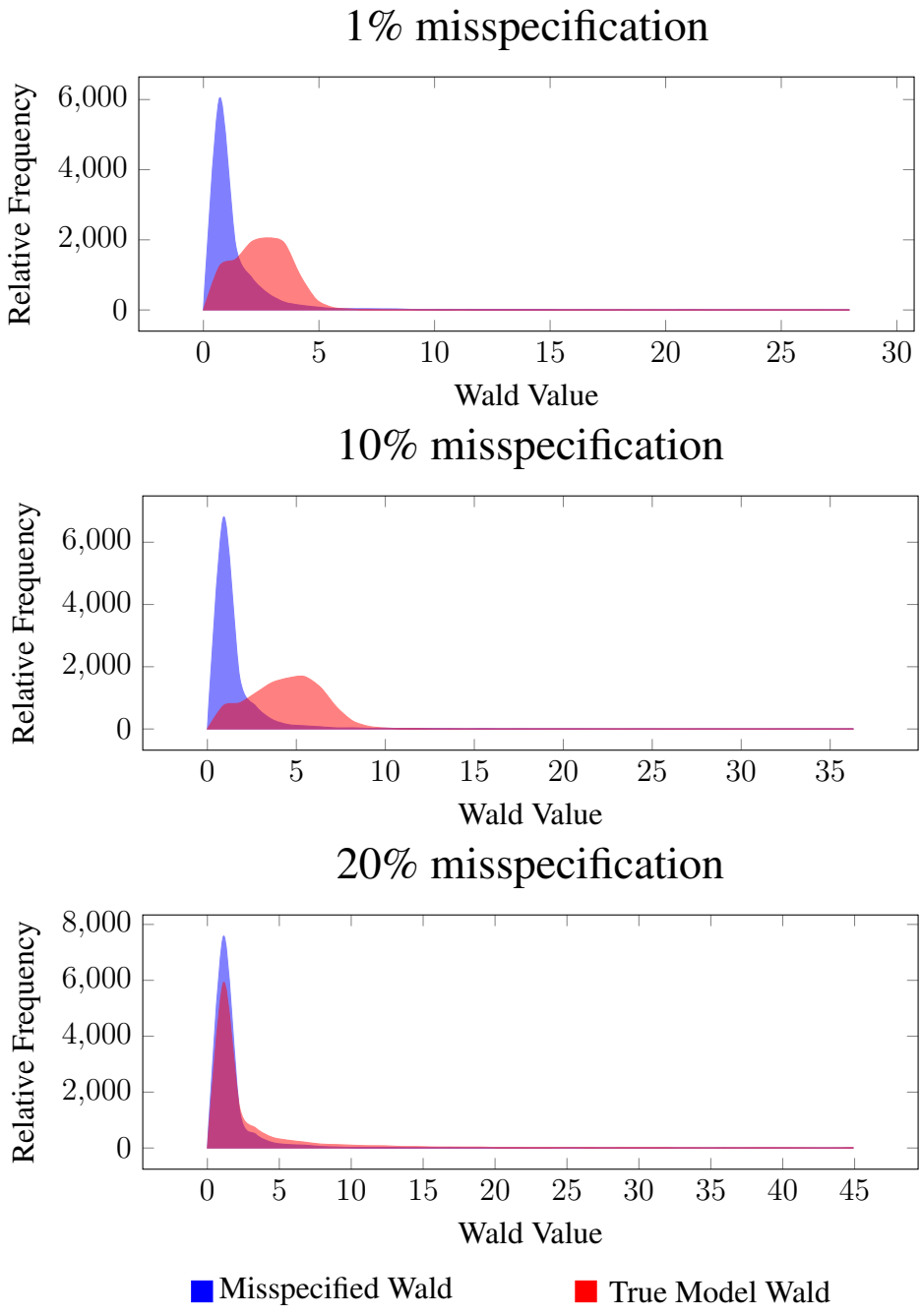


Figure 4.1: Univariate autoregressive-form parameters for 3-variable VAR, fractional integration, and simple $AR(3)$.

as much as 20% indirect inference cannot regularly distinguish the model's behaviour from that generating the data. Meanwhile, for a 7 % perturbation of the parameters, indirect inference through an $AR(1)$ actually rejects fewer instances of the false model than of the true model. This latter result suggests that output in the Smets Wouters model behaves more like a short AR process for some perturbations than for others, or for the true parameterisation. This would then mean that for certain values the variance of the false model estimates was lower relative to that for the true model, making the Wald statistics for the true parameterisation more variable. When this greater variance of the Wald statistics for the true model dominated the difference in dynamics between the true and false models, we would see the false distribution more contained within the true and a lower rejection rate:

It should be noted that this would not be possible if the parameterisation had been obtained using indirect inference estimation for the same auxiliary model, because the variance would then be minimal for the true model. But, as the parameterisation was

Figure 4.2: Wald distributions for misspecified models, and data generating model.



chosen by Bayesian estimation there is no such guarantee that the $AR(1)$ or $AR(3)$ will have maximum likelihood parameters. It would not be appropriate to repeat my experiments using true parameters estimated on an AR(1) or AR(3) auxiliary model,

as the argument above suggests the estimated values would be driven more by what these auxiliary models are capable of describing rather than the theoretical model's dynamics. Indeed, my results are a cautionary tale on the use of Indirect Inference, using an auxiliary model that cannot capture important time series properties. Figure 4.2 shows the overlap of the Wald distributions for various degrees of perturbation. This illustrates the reason for the low power, as the Wald statistics' distributions do not vary significantly with the perturbation of the tested model's parameters. Unlike Le et al. (2012) I do not here compare Indirect Inference to any other testing procedure, so statements on its performance cannot be put in context. It would be a worthwhile, but expansive, exercise to examine the implications for policy making of such poor power — that is, whether the policy recommendations were too ambiguous, or could lead to undesirable outcomes. The immediate implications of these results are relative to Indirect Inference on several variables, which actually has power.

4.6 Conclusions

This chapter set out to investigate a question tangential to the main research program of the thesis. It asked whether the technique of Indirect Inference testing could meaningfully distinguish between good and bad descriptions of data, when distantly separated data points have high correlation — that is, under the property known as long memory. It approached this question in two dimension: first, the presence of long memory in the description being assessed; second, the use of an auxiliary model that has long memory properties.

In the first dimension, it was found that indirect inference has reasonable power to reject bad descriptions of even univariate data, although its power is not fantastic for middling misspecification. Surprisingly, it was found that Indirect Inference has very low power when testing descriptions of univariate short memory data. This is an important result, as it has been used for exactly this purpose in the literature — Le et al.

(2011) for instance.

Now the second dimension, the question of long memory in the auxiliary model. Unsurprisingly, it was found that a long memory auxiliary was essential to the testing of a long memory description, unless the misspecification led to radically different qualitative behaviour. Less expected was the finding that a long memory auxiliary model can improve the power of Indirect Inference against misspecification of a short memory descriptive model.

In the context of the research program advanced by this thesis, the results in this chapter offer some support to those from the previous chapter: the hierarchical enforcement model of chapter 3 is unlikely to have been significantly misspecified, and can be considered a reasonable reductionist description of US GDP until its simplifying assumptions are successfully challenged.

Bibliography

Abadir, K. and Talmain, G. Aggregation, persistence and volatility in a macro model. *Review of Economic Studies*, 69(4):749–79, October 2002. URL http://ideas.repec.org/a/bla/restud/v69y2002i4p749-79.html.

Aloy, M. and de Truchis, G. Estimation and testing for fractional cointegration. AMSE Working Papers 1215, Aix-Marseille School of Economics, Marseille, France, June 2012. URL http://ideas.repec.org/p/aim/wpaimx/1215.html.

Bak, P., Tang, C., and Wiesenfeld, K. Self-organized criticality: An explanation of the 1/ f noise. *Physical Review Letters*, 59(4):381–384, July 1987. doi: 10.1103/physrevlett.59.381. URL http://dx.doi.org/10.1103/physrevlett.59.381.

Bak, P., Chen, K., Scheinkman, J., and Woodford, M. Aggregate fluctuations from independent sectoral shocks: self-organized criticality in a model of production and inventory dynamics. *Ricerche Economiche*, 47(1):3–30, March 1993. URL http://ideas.repec.org/a/eee/riceco/v47y1993i1p3-30.html.

Bhattacharya, R. N., Gupta, V. K., and Waymire, E. The hurst effect under trends. 1983.

Calvo, G. A. Staggered prices in a utility-maximizing framework. *Journal of Monetary Economics*, 12(3):383–398, September 1983. URL http://ideas.repec.org/a/eee/moneco/v12y1983i3p383-398.html.

Clauset, A., Shalizi, C., and Newman, M. Power-law distributions in empirical data. *SIAM Review*, 51(4):661–703, 2009. doi: 10.1137/070710111. URL http://epubs.siam.org/doi/abs/10.1137/070710111.

Dahlhaus, R. Efficient Parameter Estimation for Self-Similar Processes. *Annals of Statistics*, 17:1749–1766, 1989. doi: 10.1214/aos/1176347393.

Davidson, J. and Sibbertsen, P. Generating schemes for long memory processes: regimes, aggregation and linearity. *Journal of Econometrics*, 128(2):253 – 282, 2005. ISSN 0304-4076. doi: http://dx.doi.org/10.1016/j.jeconom.2004.08.014. URL http://www.sciencedirect.com/science/article/pii/S0304407604001563.

Erland, S. and Greenwood, P. Constructing 1/omegaalpha noise from reversible markov chains. *Phys Rev E Stat Nonlin Soft Matter Phys*, 76(3 Pt 1):031114, 2007.

Fox, R. and Taqqu, M. S. Larg-sample properties of parameter estimates for trongly dependent stationary gaussian time series. 14(2):517–532.

Geweke, J. and Porter-Hudak, S. The estimation and application of long memory time series models. *Journal of Time Series Analysis*, 4(4):221–238, 1983. ISSN 1467-9892. doi: 10.1111/j.1467-9892.1983.tb00371.x. URL http://dx.doi.org/10.1111/j.1467-9892.1983.tb00371.x.

Gil-Alana, L. A. Testing of fractional cointegration in macroeconomic time series. *Oxford Bulletin of Economics and Statistics*, 65(4):517–529, 09 2003. URL http://ideas.repec.org/a/bla/obuest/v65y2003i4p517-529.html.

Giraitis, L., Robinson, P. M., and Surgailis, D. Variance-type estimation of long memory. *Stochastic Processes and their Applications*, 80(1):1 – 24, 1999. ISSN 0304-4149. doi: http://dx.doi.org/10.1016/S0304-4149(98)00062-3. URL http://www.sciencedirect.com/science/article/pii/S0304414998000623.

Gourieroux, C., Monfort, A., and Renault, E. Indirect inference. *Journal of Applied Econometrics*, 8(S):S85–118, Suppl. De 1993. URL http://ideas.repec.org/a/jae/japmet/v8y1993isps85-118.html.

Granger, C. Long memory relationships and the aggregation of dynamic models. *Journal of Econometrics*, 14(2):227 – 238, 1980. ISSN 0304-4076. doi: http://dx.doi.org/10.1016/0304-4076(80)90092-5. URL http://www.sciencedirect.com/science/article/pii/0304407680900925.

Granger, C. W. J. and Joyeux, R. An introduction to long-memory time series models and fractional differencing. *Journal of Time Series Analysis*, 1(1):15–29, 1980. ISSN 1467-9892. doi: 10.1111/j.1467-9892.1980.tb00297.x. URL http://dx.doi.org/10.1111/j.1467-9892.1980.tb00297.x.

Gregory, A. W. and Smith, G. W. Calibration as testing: Inference in simulated macroeconomic models. *Journal of Business and Economic Statistics*, 9(3):297–303, 1991. URL http://EconPapers.repec.org/RePEc:bes:jnlbes:v:9:y:1991:i:3:p:297-303.

Grüneis, F. and Musha, T. Clustering poisson process and 1/f noise. *Japanese Journal of Applied Physics*, 25:1504, oct 1986. doi: 10.1143/JJAP.25.1504.

Hart, O. A model of imperfect competition with keynesian features. *The Quarterly Journal of Economics*, 97(1):109–38, February 1982. URL http://ideas.repec.org/a/tpr/qjecon/v97y1982i1p109-38.html.

Haubrich, J. G. and Lo, A. W. The sources and nature of long-term memory in aggregate output. *Economic Review*, (Q II):15–30, 2001. URL http://ideas.repec.org/a/fip/fedcer/y2001iqiip15-30.html.

Hurst, H. Long Term Storage Capacity of Reservoirs. *Transactions of the American Society of Civil Engineers*, 116:770–799, 1951.

Hurvich, C. M. and Chen, W. W. An Efficient Taper for Potentially Overdifferenced Long-memory Time Series. *Journal of Time Series Analysis*, 21:155–180, 2000. doi: 10.1111/1467-9892.00179.

Hurvich, C. M. and Ray, B. K. Estimation of the memory parameter for nonstationary or noninvertible fractionally integrated processes. *Journal of Time Series Analysis*, 16 (1):17–41, 1995. ISSN 1467-9892. doi: 10.1111/j.1467-9892.1995.tb00221.x. URL http://dx.doi.org/10.1111/j.1467-9892.1995.tb00221.x.

Kydland, F. E. and Prescott, E. C. Time to build and aggregate fluctuations. *Econometrica*, 50(6):1345–70, November 1982. URL http://ideas.repec.org/a/ecm/emetrp/v50y1982i6p1345-70.html.

Le, V. P. M., Meenagh, D., Minford, P., and Wickens, M. How much nominal rigidity is there in the us economy? testing a new keynesian dsge model using indirect inference. *Journal of Economic Dynamics and Control*, 35(12):2078–2104, 2011. URL http://ideas.repec.org/a/eee/dyncon/v35y2011i12p2078-2104.html.

Le, V. P. M., Meenagh, D., Minford, P., and Wickens, M. Testing dsge models by indirect inference and other methods: some monte carlo experiments. Cardiff Economics Working Papers E2012/15, Cardiff University, Cardiff Business School, Economics Section, June 2012. URL http://ideas.repec.org/p/cdf/wpaper/2012-15.html.

Lowen, S. and Teich, M. *Fractal-Based Point Processes*. Wiley Series in Probability and Statistics. Wiley, 2005. ISBN 9780471754701. URL http://books.google.co.uk/books?id=kSAaOXuJeEwC.

Lucas, R. J. Econometric policy evaluation: A critique. *Carnegie-Rochester Conference Series on Public Policy*, 1(1):19–46, January 1976. URL http://ideas.repec.org/a/eee/crcspp/v1y1976ip19-46.html.

Maddison, A. Historical statistics of the worlds economy: 1-2008 ad. URL http://www.ggdc.net/maddison/oriindex.htm.

Mandelbrot, B. Long-run linearity, locally gaussian process, h-spectra and infinite variances. *International Economic Review*, 10(1):82–111, February 1969. URL http://ideas.repec.org/a/ier/iecrev/v10y1969i1p82-111.html.

Mandelbrot, B. *The Fractal Geometry of Nature*. Henry Holt and Company, 1983. ISBN 9780716711865. URL http://books.google.co.uk/books?id=0R2LkE3N7-oC.

Mayoral, L. Further evidence on the statistical properties of real gnp. *Oxford Bulletin of Economics and Statistics*, 68(s1):901–920, December 2006. URL http://ideas.repec.org/a/bla/obuest/v68y2006is1p901-920.html.

Michelacci, C. and Zaffaroni, P. (fractional) beta convergence. *Journal of Monetary Economics*, 45(1):129–153, February 2000. URL http://ideas.repec.org/a/eee/moneco/v45y2000i1p129-153.html.

Samorodnitsky, G. Long range dependence. *Found. Trends. Stoch. Sys.*, 1(3):163–257, Jan. 2007. ISSN 1551-3106. doi: 10.1561/0900000004. URL http://dx.doi.org/10.1561/0900000004.

Shimotsu, K. Exact local whittle estimation of fractional integration with unknown mean and time trend. *Econometric Theory*, 26(02):501–540, April 2010.

Smets, F. and Wouters, R. Shocks and frictions in us business cycles: A bayesian dsge approach. *American Economic Review*, 97(3):586–606, June 2007. URL http://ideas.repec.org/a/aea/aecrev/v97y2007i3p586-606.html.

Sowell, F. Maximum likelihood estimation of stationary univariate fractionally integrated time series models. *Journal of Econometrics*, 53(1-3):165–188, 1992. URL http://ideas.repec.org/a/eee/econom/v53y1992i1-3p165-188.html.

Taylor, J. B. Staggered wage setting in a macro model. *American Economic Review*, 69(2):108–13, May 1979. URL http://ideas.repec.org/a/aea/aecrev/v69y1979i2p108-13.html.

Velasco, C. Gaussian semiparametric estimation of non-stationary time series. *Journal of Time Series Analysis*, 20(1):87–127, 1999. ISSN 1467-9892. doi: 10.1111/1467-9892.00127. URL http://dx.doi.org/10.1111/1467-9892.00127.

4.A Simulation Codes

4.A.1 Auxiliary parameter distributions for simulated data

The first of these objects generates data according to the Haubrich and Lo (2001) model, and fits an ARFIMA to these series. The second reads in external data, as produced by the Smets-Wouters implementation of Le et al. (2012), and does the same.

PercentPower.java

```
/*
 Copyright 2013 by Tom Wilkinson and Cardiff University
 Licensed under the Academic Free License version 3.0
 See the file "LICENSE" for more information
*/
package iipower;

import FELW.FELW2St;
import com.numericalmethod.suanshu.stats.timeseries.linear.
    univariate.stationaryprocess.arma.ARMAModel;
import com.numericalmethod.suanshu.stats.timeseries.linear.
    univariate.stationaryprocess.arma.ConditionalSumOfSquares;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.nio.charset.Charset;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;

/**Repeatedly simulates the Haubrich Lo (2001) model for an
    initial
```

```
* parameterisation, and then for various percentages of
    misspecification. In all
* cases, ARFIMA parameters are recorded for the generated data
    samples.
*/
public class PercentPower {
   final static Charset ENCODING = StandardCharsets.UTF_8;
   static final String OUTPUT_FILE_NAME = "iipow.csv";
   static final String OUTPUT_PATH_NAME = "D:\\simulate\\iipow
      ";

   static double[] defaultPars = {0.2,0.2,1.0};
   static double[] defaultMisSpec =
      {0.0,0.1,0.2,1.0,2.0,3.0,4.0,5.0};

   static int defaultNumARpars = 1;
   static int defaultNumMApars = 0;
   static int defaultSimLength = 5000;
   static int defaultSampleLength = 1000;
   static int defaultSimSize = 10000;
   static int defaultNumSims = 10000;

   public static void main(String[] args) {
       DateFormat dateFormat = new SimpleDateFormat("MMdd");
       Date date = new Date();

       if (keyExists("-help", args, 0))
           {
           System.err.println(
               "Format:␣␣␣␣␣␣␣␣␣␣␣java␣-jar␣
                  IndirectInferencePower.jar␣[-mode␣m]␣"+
               "[-truePars␣t]␣[-falseParfs␣f]␣[-sampleLength␣n
                  ]\n\n" +
               "-help␣␣␣␣␣␣␣␣␣␣␣␣Shows␣this␣message␣and␣exits
                  .\n\n"+
               "␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣0␣is␣Bak␣Chen␣Schenkman␣and␣
                  Woodford␣1992␣\n"+
               "␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣2␣is␣Haubrich␣and␣Lo␣2001␣\n
                  "+
               "␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣1␣is␣Davidson␣and␣Sibbertsen
                  ␣2002␣\n"+
               "-pars␣␣␣␣␣␣␣␣␣␣␣␣double[]:␣the␣min␣value␣in␣
                  the␣range␣of␣parameters␣\n\n" +
               "-numARpars␣␣␣␣␣␣␣␣double:␣the␣AR␣parameters␣of
                  ␣the␣auxiliary␣ARFIMA\n\n" +
```

```
            "-numMApars        double: the MA parameters of
                the auxiliary ARFIMA\n\n" +
            "-simLength        int > 0: the duration of the
                simulation\n\n"+
            "-sampleLength     int > 0: the size of the
                sample to take\n\n"+
            "-simSize          int > 0: the number of units
                to be aggregated\n\n"+
            "-numSims          int > 0: the number of times
                the false model \n"+
            "                  is simulated to derive the
                distribution\n\n"+
            "-misSpec          int > 0: the number of times
                the false model \n"+
            "                  is simulated to derive the
                distribution\n\n");
        System.exit(0);
        }

    java.text.NumberFormat n = java.text.NumberFormat.
        getInstance();
    n.setMinimumFractionDigits(0);
    System.err.println("IndirectInferencePower, for
        assistance type '-help' at end.");

    // set sample size
    int sampleLength = defaultSampleLength;
    String sampleLength_s = argumentForKey("-sampleLength",
        args, 0);
    if (sampleLength_s != null) {
        try
            {
            sampleLength = Integer.parseInt(sampleLength_s)
                ;
            if (sampleLength < 1) {
                    throw new Exception();
                }
            }
        catch (Exception e)
            {
            throw new RuntimeException("Invalid '
                sampleLength' value: "
                    + sampleLength_s + ", must be positive
                        integer");
            }
    }
```

```
// set sample size
int simLength = defaultSimLength;
String simLength_s = argumentForKey("-simLength", args,
    0);
if (simLength_s != null) {
    try
        {
        simLength = Integer.parseInt(simLength_s);
        if (simLength < sampleLength) {
                throw new Exception();
            }
        }
    catch (Exception e)
        {
        throw new RuntimeException("Invalid␣'simLength'
            ␣value:␣"
                + simLength_s + ",␣must␣be␣positive␣
                    integer");
        }
}

// set number of units to aggregate
int simSize = defaultSimSize;
String simSize_s = argumentForKey("-simSize", args, 0);
if (simSize_s != null) {
    try
        {
        simSize = Integer.parseInt(simSize_s);
        if (simSize < 1) {
                throw new Exception();
            }
        }
    catch (Exception e)
        {
        throw new RuntimeException("Invalid␣'simSize'␣
            value:␣"
                + simSize_s + ",␣must␣be␣positive␣
                    integer");
        }
}

// set number of simulations
int numSims = defaultNumSims;
String numSims_s = argumentForKey("-numSims", args, 0);
if (numSims_s != null) {
```

```
        try
            {
            numSims = Integer.parseInt(numSims_s);
            if (numSims < 1) {
                    throw new Exception();
                }
            }
        catch (Exception e)
            {
            throw new RuntimeException("Invalid␣'numSims'␣
                value:␣"
                    + numSims_s + ",␣must␣be␣positive␣
                        integer");
            }
    }

    // set number of AutoRegressive roots
    int numARpars = defaultNumARpars;
    String numARpars_s = argumentForKey("-numARpars", args,
        0);
    if (numARpars_s != null) {
        try
            {
            numARpars = Integer.parseInt(numARpars_s);
            if (numARpars < 0) {
                    throw new Exception();
                }
            }
        catch (Exception e)
            {
            throw new RuntimeException("Invalid␣'numARpars'
                ␣value:␣"
                    + numARpars_s + ",␣must␣be␣positive␣
                        integer");
            }
    }

    // set number of Moving Average roots
    int numMApars = defaultNumMApars;
    String numMApars_s = argumentForKey("-numMApars", args,
        0);
    if (numMApars_s != null) {
        try
            {
            numMApars = Integer.parseInt(numMApars_s);
            if (numMApars < 0) {
```

```
                    throw new Exception();
                }
            }
        catch (Exception e)
            {
            throw new RuntimeException("Invalid␣'numMApars'
                ␣value:␣"
                    + numMApars_s + ",␣must␣be␣positive␣
                        integer");
            }
    }

    // set pars
    double[] pars = defaultPars;
    for(int x=0;x<args.length-1;x++) {
        if (args[x].equalsIgnoreCase("-pars")){
            if(x+3>=args.length) {
                throw new RuntimeException("Not␣enough␣
                    arguments␣provided␣to"
                        + "-pars.␣It␣needed␣3");
            }
            for(int parIndex=0; parIndex<3; parIndex++){
                String par_s = args[x+parIndex+1];
                if (par_s != null || par_s.charAt(0) != '-'
                    ) {
                        try {
                            pars[parIndex] = Double.
                                parseDouble(par_s);
                            // we'll check elsewhere
                                whether the par values are
                                appropriate
                        }
                        catch (Exception e) {
                            throw new RuntimeException("
                                Invalid␣'-pars'␣value␣at"
                                    + "␣position␣" +
                                        parIndex + ".␣Did␣
                                        you␣provide"
                                    + 3 + "?");
                        }
                    }
                else {
                    throw new RuntimeException("Too␣few␣
                        pars␣given␣to␣'-pars'"
                        + ",␣needed␣" + 3 + ",␣but␣received␣
                            "
```

```
                        + parIndex);
                }
            }
        }
    }

    // set proportional misSpecifications
    double[] misSpec = defaultMisSpec;
    for(int x=0;x<args.length-1;x++) {
        if (args[x].equalsIgnoreCase("-misSpec")){
            ArrayList<Double> misSpecList = new ArrayList
                <>();
            int index = 0;
            String misSpec_s = args[x+1];
            while (misSpec_s != null || misSpec_s.charAt(0)
                != '-') {
                try {
                    misSpecList.add(Double.parseDouble(
                        misSpec_s));
                }
                catch (Exception e) {
                    throw new RuntimeException("Invalid␣'-
                        misSpec'␣value␣at"
                            + "␣position␣" + index);
                }
                index++;
                misSpec_s = args[x+index+1];
            }
            misSpec = new double[misSpecList.size()];
            for(index=0; index<misSpecList.size(); index++)
                {
                misSpec[index] = misSpecList.get(index).
                    doubleValue();
            }
        }
    }

    int numLMpars = numARpars + 1 + numMApars;
    int numSMpars = numARpars + numMApars;
    String directory = OUTPUT_PATH_NAME + "Haubrich01_"
            + simSize +"_"+ numARpars +"_"+ numMApars +"_"
            + dateFormat.format(date) +"\\";
    new File(directory).mkdirs();

            for(int parsIndx=0;parsIndx<misSpec.length;
                parsIndx++){
```

```
double firstBetaPar = (1.0+misSpec[parsIndx])*pars
    [0];
double secondBetaPar = (1.0+misSpec[parsIndx])*pars
    [1];
double shockVar = (1.0+misSpec[parsIndx])*pars[2];
double[][] lmParDist = new double[numSims][
    numLMpars];
double[][] smParDist = new double[numSims][
    numSMpars];

/* Now, for the simulations */
for(int sim=0; sim<numSims; sim++){
    double[] simData = IndirectInferencePower.
        haubrich01(simLength,simSize,
            firstBetaPar, secondBetaPar, shockVar);
    double[] sample = new double[sampleLength];
    int time = simLength - sampleLength;
    for(int obs=0;obs<sampleLength;obs++){
        sample[obs] = simData[time];
        time++;
    }
    double diffPar = new
            FELW2St(sample,(int)Math.floor(Math.pow
                (sampleLength,0.65)),3)
                    .estimate();
    // estimte ARMAs for both the differenced and
        undifferenced
    // data
    ARMAModel smARMA = new
            ConditionalSumOfSquares(sample,
                numARpars,0,numMApars)
            .getARMAModel();
    ARMAModel lmARMA = new
            ConditionalSumOfSquares(FELW2St.
                fracdiff(sample
            ,diffPar),numARpars,0,numMApars).
                getARMAModel();

    /* first we'll record the AR parameters for the
        short- and
     * long- memory auxiliary models */
    int distIndx;
    for(int ARpar=0; ARpar<numARpars; ARpar++){
        lmParDist[sim][ARpar] = lmARMA.AR(ARpar+1);
        smParDist[sim][ARpar] = smARMA.AR(ARpar+1);
    }
```

```
            /* Now to deal with the differencing parameter
               in the long-
             * memory auxiliary model */
            lmParDist[sim][numARpars] = diffPar;
            /* finally come the MA parameters for the short
               - and long-
             * memory auxiliary models */
            for(int MApar=0; MApar<numMApars; MApar++){
                int arrayIndx = MApar + numARpars;
                lmParDist[sim][arrayIndx+1] = lmARMA.MA(
                    MApar+1);
                smParDist[sim][arrayIndx] =  smARMA.MA(
                    MApar+1);
            }
        }
        try{
            // the output file needs to be named so as to
               be uniquely identified
            // (hence the date), but give relevant
               information quickly (hence params)
            Path outputPath = Paths.get(directory +
               firstBetaPar
                    + "_" + secondBetaPar + "_" + shockVar
                       + "_"
                    + OUTPUT_FILE_NAME);
            File outputFile = new File(outputPath.toString
               ());
            outputFile.createNewFile();

            try(BufferedWriter writer = Files.
               newBufferedWriter(outputPath,
                    ENCODING)){
                // Let's print out the whole parameter list
                writer.write("Economic␣model␣parameters:␣
                   firstBetaPar␣"
                        + firstBetaPar + ",␣secondBetaPar␣"
                        + secondBetaPar + ",␣shockVar␣" +
                            shockVar);
                writer.newLine();
                writer.write(numARpars + ",␣" + numMApars
                        + ",␣Long-memory␣auxiliary␣model:␣
                            ARIMA("+numARpars
                        +",␣d,␣"+numMApars+");␣␣"
                        + "Short-memory␣auxiliary␣model:␣
                            ARMA("+numARpars
                        +",␣"+numMApars+")");
```

```
                    writer.newLine();
                    writer.write("Distribution(s)...");
                    writer.newLine();
                    // Now the distributions themselves
                    for(int sim=0; sim<numSims; sim++){
                        for(int ARpar=0; ARpar<numARpars; ARpar
                            ++)
                            writer.write(lmParDist[sim][ARpar]
                                +",␣");
                        writer.write(lmParDist[sim][numARpars]+
                            ",␣");
                        for(int MApar=numARpars+1; MApar<
                            numLMpars; MApar++)
                            writer.write(lmParDist[sim][MApar]+
                                ",␣");
                        for(int ARpar=0; ARpar<numARpars; ARpar
                            ++)
                            writer.write(smParDist[sim][ARpar]+
                                ",␣");
                        for(int MApar=numARpars; MApar<
                            numSMpars; MApar++)
                            writer.write(smParDist[sim][MApar]+
                                ",␣");
                        writer.newLine();
                    }
                }
            }
            catch (FileNotFoundException e)
            {
                System.err.println("FileNotFoundException:␣" +
                    e.getMessage());
            }
            catch (IOException e)
            {
                System.err.println("Caught␣IOException:␣" + e.
                    getMessage());
            }
        }

    }

    static String argumentForKey(String key, String[] args, int
        startingAt)
        {
        for(int x=0;x<args.length-1;x++)  // key can't be the
            last string
```

```
            if (args[x].equalsIgnoreCase(key))
                return args[x + 1];
        return null;
        }

    static boolean keyExists(String key, String[] args, int
       startingAt)
        {
        for(int x=0;x<args.length;x++)  // key can't be the
           last string
            if (args[x].equalsIgnoreCase(key))
                return true;
        return false;
        }
}
```

ExtData.java

```
/*
 Copyright 2013 by Tom Wilkinson and Cardiff University
 Licensed under the Academic Free License version 3.0
 See the file "LICENSE" for more information
*/
package iipower;

import FELW.FELW2St;
import com.numericalmethod.suanshu.stats.timeseries.linear.
   univariate.stationaryprocess.arma.ARMAModel;
import com.numericalmethod.suanshu.stats.timeseries.linear.
   univariate.stationaryprocess.arma.ConditionalSumOfSquares;
import java.io.BufferedWriter;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.nio.charset.Charset;
import java.nio.charset.StandardCharsets;
import java.nio.file.DirectoryStream;
import java.nio.file.FileSystems;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.Iterator;
```

```
/**Reads simulation data from the files in a given directory,
   and fits an
* auxiliary ARFIMA model to them, recording the parameter
   distributions.
*/
public class ExtData {
   static Path inputFolder;
   final static Charset ENCODING = StandardCharsets.UTF_8;
   static final String OUTPUT_FILE_NAME = "iipow.csv";
   static final String OUTPUT_PATH_NAME = "D:\\simulate\\iipow
      ";

   static String defaultDirName = "D:\\simulate\\iiSWy\\y\\";

   static int defaultNumARpars = 1;
   static int defaultNumMApars = 0;
   static int defaultNumSims = 10000;
   static double defaultParDeviation = 0.0;

   public static void main(String[] args) throws IOException {
       DateFormat dateFormat = new SimpleDateFormat("MMdd");
       Date date = new Date();

       if (keyExists("-help", args, 0))
           {
           System.err.println(
               "Format: ␣␣␣␣␣␣␣␣␣␣␣java ␣-jar ␣
                  IndirectInferencePower.jar ␣[-mode ␣m]␣"
                   +                "[-truePars ␣t]␣[-falseParfs
                      ␣f]␣[-sampleLength ␣n]\n\n" +
               "-help ␣␣␣␣␣␣␣␣␣␣␣␣Shows ␣this ␣message ␣and ␣exits
                  .\n\n" +
               "-dirName ␣␣␣␣␣␣␣␣␣string: ␣the ␣location ␣of ␣the ␣
                  simulation ␣.CSVs ␣\n\n" +
               "-parDeviation ␣␣␣␣␣double: ␣for ␣record ␣the ␣
                  percentage ␣deviation ␣\n\n" +
               "-numARpars ␣␣␣␣␣␣␣␣double[]: ␣the ␣AR ␣parameters ␣
                  of ␣the ␣auxiliary ␣ARFIMA\n\n" +
               "-numMApars ␣␣␣␣␣␣␣␣double[]: ␣the ␣MA ␣parameters ␣
                  of ␣the ␣auxiliary ␣ARFIMA\n\n" +
               "-numSims ␣␣␣␣␣␣␣␣␣␣int ␣>␣0: ␣the ␣number ␣of ␣times
                  ␣the ␣false ␣model ␣\n"
             + "␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣is ␣simulated ␣to ␣derive ␣the ␣
                  distribution\n\n");
           System.exit(0);
```

```
    }

java.text.NumberFormat n = java.text.NumberFormat.
    getInstance();
n.setMinimumFractionDigits(0);
System.err.println("IndirectInferencePower␣for␣
    assistance␣type␣'␣-help'␣at␣end.");

// set dirName
String dirName = defaultDirName;
String dirName_s = argumentForKey("-dirName", args, 0);
if (dirName_s != null) {
    dirName = dirName_s;
}

// set parDeviation
double parDeviation = defaultParDeviation;
String parDeviation_s = argumentForKey("-parDeviation",
    args, 0);
if (parDeviation_s != null) {
    try
        {
        parDeviation = Double.parseDouble(
            parDeviation_s);
        if (parDeviation < 0.0 || parDeviation > 100.0)
            {
                throw new Exception();
            }
        }
    catch (Exception e)
        {
        throw new RuntimeException("Invalid␣'
            parDeviation'␣value:␣"
                + parDeviation_s + "␣must␣be␣a␣double␣
                    in␣[0.0,100.0]");
        }
}

// set number of simulations
int numSims = defaultNumSims;
String numSims_s = argumentForKey("-numSims", args, 0);
if (numSims_s != null) {
    try
        {
        numSims = Integer.parseInt(numSims_s);
        if (numSims < 1) {
```

```
                    throw new Exception();
                }
            }
        catch (Exception e)
            {
            throw new RuntimeException("Invalid␣'numSims'␣
                value:␣"
                    + numSims_s + ",␣must␣be␣positive␣
                        integer");
            }
    }

    // set number of AutoRegressive roots
    int numARpars = defaultNumARpars;
    String numARpars_s = argumentForKey("-numARpars", args,
        0);
    if (numARpars_s != null) {
        try
            {
            numARpars = Integer.parseInt(numARpars_s);
            if (numARpars < 0) {
                    throw new Exception();
                }
            }
        catch (Exception e)
            {
            throw new RuntimeException("Invalid␣'numARpars'
                ␣value:␣"
                    + numARpars_s + ",␣must␣be␣positive␣
                        integer");
            }
    }

    // set number of Moving Average roots
    int numMApars = defaultNumMApars;
    String numMApars_s = argumentForKey("-numMApars", args,
        0);
    if (numMApars_s != null) {
        try
            {
            numMApars = Integer.parseInt(numMApars_s);
            if (numMApars < 0) {
                    throw new Exception();
                }
            }
        catch (Exception e)
```

```
                {
                throw new RuntimeException("Invalid␣'numMApars'
                    ␣value:␣"
                        + numMApars_s + ",␣must␣be␣positive␣
                            integer");
                }
        }


        // there seems little point creating an instance of the
            class; the test
        // is simple enough to be run here as if procedural
//          IndirectInferencePower powerTest = new
    IndirectInferencePower();

        int numLMpars = numARpars + 1 + numMApars;
        int numSMpars = numARpars + numMApars;
        String directory = OUTPUT_PATH_NAME + "extData" + "_"
                + dateFormat.format(date) + "\\";
        new File(directory).mkdirs();

        double[][] lmParDist = new double[numSims][numLMpars];
        double[][] smParDist = new double[numSims][numSMpars];

        /* Now to business: we need to iterate through the
            files in the directory */
        /* BUT we want a filter to ensure we only look at .txt
            files */
        DirectoryStream.Filter<Path> filter = new
            DirectoryStream.Filter<Path>() {
             @Override
               public boolean accept(Path file) throws
                   IOException {
                    return (file.getName(file.getNameCount()-1).
                        toString()
                            .endsWith(".csv"));
               }
          };
        inputFolder = Paths.get(dirName);
        try(DirectoryStream<Path> dirStream
                = Files.newDirectoryStream(FileSystems.
                    getDefault()
                .getPath(dirName),filter)){
            // now to get the first run of simulated data and
                initialise the
```

```
// ArrayList to avoid null pointer exceptions in
   the ordering
// loop
double[] sample;
int sampleLength;
Iterator<Path> dirIterator = dirStream.iterator();
/* Now, for each of the subsequent simulations, the
    parameter
 * estimates will be added to the ArrayLists in
    ascending order */
int sim = 0;
while(dirIterator.hasNext()){
    Path file = dirIterator.next();
    ArrayList<Double> timeSeries = new ArrayList
       <>();
    try (BufferedReader reader
            = Files.newBufferedReader(file,
               ENCODING)){
        String line = null;
        /* we read through the remaining lines...
           */
        while((line = reader.readLine())!=null){
            timeSeries.add(Double.parseDouble(line)
               );
        }
    }
    sampleLength = timeSeries.size();
    sample = new double[sampleLength];
    for(int time=0; time<sampleLength; time++){
        sample[time] = timeSeries.get(time).
           doubleValue();
    }
    double diffPar = new
            FELW2St(sample,(int)Math.floor(Math.pow
               (sampleLength,0.65)),3)
                    .estimate();
    // estimte ARMAs for both the differenced and
       undifferenced
    // data
    ARMAModel smARMA = new
            ConditionalSumOfSquares(sample,
               numARpars,0,numMApars)
            .getARMAModel();
    ARMAModel lmARMA = new
            ConditionalSumOfSquares(FELW2St.
               fracdiff(sample,diffPar),
```

```
                                numARpars,0,numMApars).
                                    getARMAModel();

                /* first we'll record the AR parameters for the
                    short- and
                 * long- memory auxiliary models */
                for(int ARpar=0; ARpar<numARpars; ARpar++){
                    lmParDist[sim][ARpar] = lmARMA.AR(ARpar+1);
                    smParDist[sim][ARpar] = smARMA.AR(ARpar+1);
                }
                /* Now to deal with the differencing parameter
                    in the long-
                 * memory auxiliary model */
                lmParDist[sim][numARpars] = diffPar;
                /* finally come the MA parameters for the short
                    - and long-
                 * memory auxiliary models */
                for(int MApar=0; MApar<numMApars; MApar++){
                    int arrayIndx = MApar + numARpars;
                    lmParDist[sim][arrayIndx+1] = lmARMA.MA(
                        MApar+1);
                    smParDist[sim][arrayIndx] = smARMA.MA(MApar
                        +1);
                }
                sim++;
            }
        }

        try{
            // the output file needs to be named so as to be
                uniquely identified
            // (hence the date), but give relevant information
                quickly (hence params)
            Path outputPath = Paths.get(directory +
                parDeviation + "_"
                    + OUTPUT_FILE_NAME);
            File outputFile = new File(outputPath.toString());
            outputFile.createNewFile();

            try(BufferedWriter writer = Files.newBufferedWriter
                (outputPath,
                    ENCODING)){
                // Let's print out the whole parameter list
                writer.write("Economic␣model␣parameters␣deviate
                    ␣by␣=␣"
                        + parDeviation + "␣percent");
```

```
                writer.newLine();
                writer.write(numARpars + ", " + numMApars
                        + ", Long-memory auxiliary model: ARIMA
                            ("+numARpars
                        +", d, "+numMApars+");  "
                        + "Short-memory auxiliary model: ARMA("
                            +numARpars
                        +", "+numMApars+")");
                writer.newLine();
                writer.write("Distribution(s)...");
                writer.newLine();
                // Now the distributions themselves
                for(int sim=0; sim<numSims; sim++){
                    for(int ARpar=0; ARpar<numARpars; ARpar++)
                        {
                        writer.write(lmParDist[sim][ARpar] +",
                            ");
                    }
                    writer.write(lmParDist[sim][numARpars]+",
                        );
                    for(int MApar=numARpars+1; MApar<numLMpars;
                        MApar++) {
                        writer.write(lmParDist[sim][MApar]+",
                            );
                    }
                    for(int ARpar=0; ARpar<numARpars; ARpar++)
                        {
                        writer.write(smParDist[sim][ARpar]+",
                            );
                    }
                    for(int MApar=numARpars; MApar<numSMpars;
                        MApar++) {
                        writer.write(smParDist[sim][MApar]+",
                            );
                    }
                    writer.newLine();
                }
            }
        }
        catch (FileNotFoundException e)
        {
            System.err.println("FileNotFoundException: " + e.
                getMessage());
        }
        catch (IOException e)
        {
```

```
            System.err.println("Caught␣IOException:␣" + e.
                getMessage());
        }
    }

    static String argumentForKey(String key, String[] args, int
        startingAt){
        // key can't be the last string
        for(int x=0;x<args.length-1;x++){
            if (args[x].equalsIgnoreCase(key)){
                return args[x + 1];
            }
        }
        return null;
    }

    static boolean keyExists(String key, String[] args, int
       startingAt){
        // key can't be the last string
        for(int x=0;x<args.length;x++)  {
            if (args[x].equalsIgnoreCase(key)){
                return true;
            }
        }
        return false;
    }
}
```

4.A.2 Comparison of Wald distributions

IIPowerCompare.java

```
/*
 Copyright 2013 by Tom Wilkinson and Cardiff University
 Licensed under the Academic Free License version 3.0
 See the file "LICENSE" for more information
*/
package iipowercompare;
import java.io.*;
import java.nio.charset.Charset;
import java.nio.charset.StandardCharsets;
import java.nio.file.*;
import java.util.ArrayList;
import org.ojalgo.matrix.BasicMatrix;
import org.ojalgo.matrix.PrimitiveMatrix;
```

```
/**Assesses the power of indirect inference on various models:
* - simulated distribution of Auxiliary parameter values is
    read in for
*    true model
* the following is repeated for each percentage deviation:
* - simulated distributions for false model aux-par
    distributions are read
* - means and covariance matrices are calculated for false
    model aux-par
*    distributions
* - the WALD values are found for both distributions, and the
    95-percentiles
*    calculated for the False distribution
* - the number of true model Wald stats that fall outside the
    95 percentile
*    for the false model are counted; this is the probability
    that the false
*    model is rejected according to the L2 norm
*
* WARNING: this is not suitable for auxiliary models with MA
    parameters */
public class Iipowercompare {
    final static double SIGNIFICANCE = 0.95;
    final static Charset ENCODING = StandardCharsets.UTF_8;
    static final String INPUT_PATH = "C:\\users\\tom\\"
            + "dropbox\\iiHLy\\ar3long\\";
    static final String FALSE_MODEL_PATTERN = "_iipow.csv";
    static final String TRUE_MODEL_DATA = "0.0_iipow.csv";
    static final String OUTPUT_FILE = "iipow.csv";
    static final String OUTPUT_PATH = "C:\\users\\tom\\"
            + "dropbox\\iiHLy\\ar3long\\";

    public static void main(String[] args) throws IOException {
        /* the distribution of the parameters should be read in
            for the true
         * model - for both the short memory and long memory
            auxilliary models */
        double[][] trueLMParDist;
        double[][] trueSMParDist;
        int numARpars;
        int numMApars;
        int numPars;
        int numTrueObs;
        Path truePath = FileSystems.getDefault().getPath(
            INPUT_PATH,TRUE_MODEL_DATA);
```

```
class DoubleArrayList extends ArrayList<Double> { }

try (BufferedReader reader
        = Files.newBufferedReader(truePath, ENCODING)){
    // skip the parameter list and stats
    System.out.println(reader.readLine());
    /* read in the numbers of AR and MA pars */
    String line = reader.readLine();
    String[] brokenLine = line.split(",␣");
    numARpars = Integer.parseInt(brokenLine[0]);
    numMApars = Integer.parseInt(brokenLine[1]);
    numPars = numARpars + numMApars;
    /* skip the "distributions..." line */
    System.out.println(reader.readLine());
    System.out.println(reader.readLine());
    System.out.println(reader.readLine());
    System.out.println(reader.readLine());
    System.out.println(reader.readLine());

    // now to read the distribution from the file into
       a list
    ArrayList<String> parList = new ArrayList<>();
    while((line = reader.readLine()) != null){
        parList.add(line);
    }
    numTrueObs = parList.size();
    trueLMParDist = new double[numTrueObs][numPars+1];
    trueSMParDist = new double[numTrueObs][numPars];
    for(int obs=0; obs<numTrueObs; obs++){
        brokenLine = parList.get(obs).split(",␣");
        for(int par=0; par<numPars; par++){
            trueLMParDist[obs][par] = Double.
               parseDouble(brokenLine[par]);
            trueSMParDist[obs][par]
                    = Double.parseDouble(brokenLine[
                       numPars + 1 + par]);
        }
        /* now the long memory parameter */
        trueLMParDist[obs][numPars] = Double.
           parseDouble(brokenLine[numPars]);
    }
}

/* we'll output to a single file for all of the false
   models read in */
Path outPath
```

```
            = FileSystems.getDefault().getPath(OUTPUT_PATH,
                OUTPUT_FILE);
        try(BufferedWriter writer = Files.newBufferedWriter(
            outPath, ENCODING)){
            /* write a header for the file */
            writer.write("numARpars = "+ numARpars +", 
                numMApars = "+ numMApars);
            writer.newLine();
            writer.write("percentage deviation; long memory aux
                model, rejection "
                    + "rate using Wald norm; long memory aux
                        model, rejection "
                    + "rate using infty norm; short memory aux
                        model, "
                    + "rejection rate using Wald norm; short
                        memory aux "
                    + "model, rejection rate using infty norm."
                        );
            writer.newLine();

            // we need a filter to make sure only .CSVs of the
                right kind are read in
            DirectoryStream.Filter<Path> filter = new
                DirectoryStream.Filter<Path>() {
                    public boolean accept(Path file) throws
                        IOException {
                        return (file.getName(file.getNameCount()
                            -1).toString()
                                .endsWith(FALSE_MODEL_PATTERN));
                    }
            };
            /* Now to check, for every false model aux-par
                distribution, how
             * many of the true model simulations are rejected
                */
            try(DirectoryStream<Path> dirStream
                = Files.newDirectoryStream(FileSystems.
                    getDefault()
                .getPath(INPUT_PATH), filter)){
                for(Path falsePath : dirStream){
                    /* need the first part of the filename in
                        order to keep
                     * track of the percentage falseness */
                    double parDeviation = Double.parseDouble(
                        falsePath
```

```
            .getFileName().toString().split("_"
                )[0]);

        int numFalseObs;
        double[][] falseLMParDist;
        double[][] falseSMParDist;
        double[] falseLMMean;
        double[] falseSMMean;
        double[][] falseLMCov;
        double[][] falseSMCov;
        ArrayList<Double> falseLMWaldDist;
        ArrayList<Double> falseSMWaldDist;
        double propLMRejectWald;
        double propSMRejectWald;
        /* now to read in the distribution */
        try (BufferedReader reader
                = Files.newBufferedReader(falsePath
                    , ENCODING)){
            /* skip the parameter list */
            for(int i=0; i<7; i++){
                System.out.println(reader.readLine
                    ());
            }
            String line = null;
            String[] brokenLine = null;
            // now to read the distribution from
                the file into a list
            ArrayList<String> parList = new
                ArrayList<>();
            while((line = reader.readLine()) !=
                null){
                parList.add(line);
            }

            /* read in the distributions and
                calculate stats */
            numFalseObs = parList.size();
            falseLMParDist = new double[numFalseObs
                ][numPars+1];
            falseSMParDist = new double[numFalseObs
                ][numPars];
            falseLMMean = new double[numPars+1];
            falseSMMean = new double[numPars];
            falseLMCov = new double[numPars+1][
                numPars+1];
```

```
falseSMCov = new double[numPars][
    numPars];
for(int obs=0; obs<numFalseObs; obs++){
    brokenLine = parList.get(obs).split
        (",␣");
    for(int par=0; par<numPars; par++){
        double lmPar = Double.
            parseDouble(brokenLine[par])
            ;
        double smPar = Double
                .parseDouble(brokenLine
                    [numPars + 1 + par])
                    ;
        falseLMParDist[obs][par] =
            lmPar;
        falseSMParDist[obs][par] =
            smPar;

        /* the contribution to means */
        falseLMMean[par] += lmPar;
        falseSMMean[par] += smPar;

        /* the contribution to
            covariances*/
        for(int otherPar=0; otherPar<=
            par; otherPar++){
            falseLMCov[par][otherPar]
                    += lmPar*
                        falseLMParDist[
                        obs][otherPar];
            falseSMCov[par][otherPar]
                    += smPar*
                        falseSMParDist[
                        obs][otherPar];
        }
    }
    /* now the long memory difference
        parameter
     * NOTE: this is broken if there
        are MA pars */
    double diffPar = Double.parseDouble
        (brokenLine[numPars]);
    falseLMParDist[obs][numPars] =
        diffPar;
    falseLMCov[numPars][numPars] +=
        diffPar*diffPar;
```

```
            for(int otherPar=0; otherPar<
                numPars; otherPar++){
                 falseLMCov[numPars][otherPar]
                         += diffPar*
                             falseLMParDist[obs][
                             otherPar];
            }
        }
    }
    /* now we finish the stats */
    falseLMMean[numPars] /= numFalseObs;
    falseLMCov[numPars][numPars] /= numFalseObs
        ;
    falseLMCov[numPars][numPars]
            -= falseLMMean[numPars]*falseLMMean
                [numPars];
    for(int par=0; par<numPars; par++){
        falseLMMean[par] /= numFalseObs;
        falseSMMean[par] /= numFalseObs;

        for(int otherPar=0; otherPar<par;
            otherPar++){
             falseLMCov[par][otherPar] /=
                 numFalseObs;
             falseLMCov[par][otherPar]
                     -= falseLMMean[par]*
                         falseLMMean[otherPar];
             falseLMCov[otherPar][par] =
                 falseLMCov[par][otherPar];
             falseSMCov[par][otherPar] /=
                 numFalseObs;
             falseSMCov[par][otherPar]
                     -= falseSMMean[par]*
                         falseSMMean[otherPar];
             falseSMCov[otherPar][par] =
                 falseSMCov[par][otherPar];
        }
        falseLMCov[numPars][par] /= numFalseObs
            ;
        falseLMCov[numPars][par]
                -= falseLMMean[par]*falseLMMean
                    [numPars];
        falseLMCov[par][numPars] = falseLMCov[
            numPars][par];
    }
```

```
/* now to invert the covariance matrices */
final BasicMatrix.Factory<?> matrixFactory
        = PrimitiveMatrix.FACTORY;
BasicMatrix invCovLM = matrixFactory.rows(
    falseLMCov)
        .invert();
BasicMatrix invCovSM = matrixFactory.rows(
    falseSMCov)
        .invert();

/* and create the vactors of means for
    later */
BasicMatrix meanLM = matrixFactory.rows(
    falseLMMean);
BasicMatrix meanSM = matrixFactory.rows(
    falseSMMean);

/* now the preparations before building the
     Wald
 * distributions */
falseLMWaldDist = new ArrayList<>();
falseSMWaldDist = new ArrayList<>();
/* To order later, we need to populate with
     the first */
BasicMatrix centralisedPars
        = matrixFactory.rows(falseLMParDist
            [0])
        .subtract(meanLM);
double falseWald
        = invCovLM.multiplyLeft(
            centralisedPars)
        .multiplyVectors(centralisedPars).
            doubleValue();
falseLMWaldDist.add(falseWald);
centralisedPars
        = matrixFactory.rows(falseSMParDist
            [0])
        .subtract(meanSM);
falseWald
        = invCovSM.multiplyLeft(
            centralisedPars)
        .multiplyVectors(centralisedPars).
            doubleValue();
falseSMWaldDist.add(falseWald);
```

```
/* now subsequent values can be compared to
    these */
for(int obs=1; obs<numFalseObs; obs++){
    /* Build up the Wald Distributions */
    centralisedPars
            = matrixFactory.rows(
                falseLMParDist[obs])
            .subtract(meanLM);
    falseWald
            = invCovLM.multiplyLeft(
                centralisedPars)
            .multiplyVectors(
                centralisedPars).doubleValue
                ();
    boolean ordered = false;
    for(int ordrdObs=0;
            ordrdObs<falseLMWaldDist.size()
                ;
            ordrdObs++){
        if(falseWald<falseLMWaldDist.get(
            ordrdObs)){
            falseLMWaldDist.add(ordrdObs,
                new Double(falseWald));
            ordered = true;
            break;
        }
    }
    if(ordered==false){
        falseLMWaldDist.add(new Double(
            falseWald));
    }
    centralisedPars
            = matrixFactory.rows(
                falseSMParDist[obs])
            .subtract(meanSM);
    falseWald
            = invCovSM.multiplyLeft(
                centralisedPars)
            .multiplyVectors(
                centralisedPars).doubleValue
                ();
    ordered = false;
    for(int ordrdObs=0;
            ordrdObs<falseSMWaldDist.size()
                ;
            ordrdObs++){
```

```
                if(falseWald<falseSMWaldDist.get(
                    ordrdObs)){
                    falseSMWaldDist.add(ordrdObs,
                        new Double(falseWald));
                    ordered = true;
                    break;
                }
            }
            if(ordered==false){
                falseSMWaldDist.add(new Double(
                    falseWald));
            }
        }
        /* now count the proportion of true
            simulations that
         * fall in the rejection region of Wald
            distribution */
        double alphaWald = (1.0 - SIGNIFICANCE);
        int uBoundWaldIndx
                = (int)Math.round((1.0-alphaWald)*(
                    double)numFalseObs);
        double uBoundWaldLM = falseLMWaldDist.get(
            uBoundWaldIndx).doubleValue();
        double uBoundWaldSM = falseSMWaldDist.get(
            uBoundWaldIndx).doubleValue();
        int numLMReject = 0;
        int numSMReject = 0;
        for(int obs=0; obs<numTrueObs; obs++){
            double trueWald;
            centralisedPars
                    = matrixFactory.rows(
                        trueLMParDist[obs])
                    .subtract(meanLM);
            trueWald
                    = invCovLM.multiplyLeft(
                        centralisedPars)
                    .multiplyVectors(
                        centralisedPars).doubleValue
                        ();
            if(trueWald>uBoundWaldLM){
                numLMReject++;
            }
            centralisedPars
                    = matrixFactory.rows(
                        trueSMParDist[obs])
                    .subtract(meanSM);
```

```
                        trueWald
                                = invCovSM.multiplyLeft(
                                    centralisedPars)
                                .multiplyVectors(
                                    centralisedPars).doubleValue
                                    ();
                        if(trueWald>uBoundWaldSM){
                            numSMReject++;
                        }
                    }
                    propLMRejectWald = (double)numLMReject / (
                        double)numTrueObs;
                    propSMRejectWald = (double)numSMReject / (
                        double)numTrueObs;


                    /* now to write these to the output file */
                    writer.write(parDeviation +"\t"+
                        propLMRejectWald +"\t"+
                            propSMRejectWald);
                    writer.newLine();
                }
            }
            catch (IOException e)
            {
                System.err.println("Caught␣IOException:␣" + e.
                    getMessage());
            }
        }
        catch (FileNotFoundException e)
        {
            System.err.println("FileNotFoundException:␣" + e.
                getMessage());
        }
        catch (IOException e)
        {
            System.err.println("Caught␣IOException:␣" + e.
                getMessage());
        }
    }
}
```

4.A.3 Estimation of the fractional difference parameter

See FELW2St.java in appendix 3.A.3.

4.A.4 Utilities used in this testing

See MersenneTwisterFast.java in appendix 3.A.3.

Chapter 5

Limits on the Formation of Small World Networks by Strategic Action

5.1 Introduction

The preceding chapters 3 and 4 established a model of macroeconomic volatility based not on exchange, but rather on the enforcement of agreements between individuals. The aggregate volatility came about in the context of hierarchies of third party enforcers, putting just a few high up enforcers in a position to influence the productivity of many. Recessions were then the product of successive failure to enforce of whole avalanches of these enforcers. But, chapters 1 and 2 had called for more than just new models of macroeocnomic volatility. They had shown that another important macroeconomic phenomenon also remained unsatisfactorily modelled, namely the differences in growth between different economies. In chapter 1 I sketched an intuitive model by which this phenomenon too could arise through considerations of enforcement alone. But, in this case, it wasn't only the hierarchical third-party enforcement that was needed. I proposed that third-party enforcement would coexist and compete with peer-pressure based enforcement. This latter mechanism would rely on the value of future interaction with a community of individuals to act as collateral for present interactions between those individuals. In terms of growth, my supposition was that dense communities, established for the sake of providing this collateral, could prevent

the development of the hierarchical enforcement even when advancing technology had made the latter preferable. Without the formal legal frameworks of that hierarchical enforcement, I suggested that long distance trade and Economic growth would be stifled, thereby explaining the divergence of Western economies from formerly comparable economies elsewhere. In order to proceed in modelling this interaction of peer-pressure and hierarchy based enforcement, we first need to establish models of peer-pressure enforcement alone. This chapter is a starting point along that path.

As proposed in Chapters 1 and 3, the growing work on agent-networks, and the dynamics of complex systems[1], more generally offers an opportunity to reunify the study of society, against Pareto's split into the dubiously orthogonal Economics and Sociology. Towards this goal too, of predictively useful socio-economic models, this chapter contributes to answering the question of how observed social network structure might arise from the behaviour of its constituent agents.

The particular structural feature on which I focus, for both enforcement and more general social structure, is the "small worlds" property named for the famous experiments of Travers and Milgram (1969). The popular concept of "six degrees of separation" captures the Small World property: imagine I should pick another person at random, and pass a message for them to an acquaintance of mine, and that acquaintance to an acquaintance of their own etc.; I might describe the society as a *small world* if relatively few such passes are needed for the message to reach its recipient. Formally, we call the vector of acquaintance relationships a *path*, and the number of passes along it the *path length*. Consider a *network generating mechanism*, that picks the relationships for a social network within a previously unconnected population; the small worlds property just described will be related to the shortest paths between each pair of agents in the generated network. For a population of millions to have these "six degrees of separation", it is obvious that the shortest path lengths must have increased, relative to the population, at a sub-linear rate from the minimal population of just two agents,

[1] *Complex System* is usually taken to denote a collection of adaptive agents, with *emergent* collective behaviour (different from that of the agents) arising in a distributed way (Boccara, 2004).

with *one degree of separation*. Using the average to describe the set of these shortest path length, we then say that the mechanism has the Small Worlds property if, as we increase the size of population to which it is applied, the average grows at the same speed as the natural logarithm of the population size.

Observed social networks often tend to exhibit both the small worlds property and high *clustering* —which is the tendency of two neighbouring nodes to share a common neighbour, a kind of *density* to the graph (Jackson, 2010). These are, in a sense, opposite properties, with one requiring minimal closed paths within the network (being wastes of links that could reach new nodes) and the other requiring as many closed paths of length three as possible. This presents an obvious problem for strategic models, where the behaviour of individual agents must therefore include drives for both short closed paths (or *cycles*) and long open paths. Nevertheless, Methodological Individualist strategic models[2] are essential where one wishes to predict changes to social network topology arising from unprecedented changes to agents or the environment. The focus of this chapter is, therefore, the potential for strategic behaviour to promote the Small Worlds property, in the face of a constant pressure towards clustering. In order to simplify the analysis, the pressure towards clustering is non-strategic, being imposed through the search mechanism by which potential relationships are available for strategic consideration.

The topological peculiarity of combined high clustering and small worlds is intimately related to the intuitive growth model I sketched above and in chapter 1. Peer pressure enforcement demands that the two parties to an agreement each have agreements with many of the same individuals, as per the *closure* of Coleman (1988). Only then can the future value of all those agreements act as collateral to any one agreement until it is satisfied. In a network of agreements, this sharing of counterparts is described by high clustering. Conversely, regardless of whether their agreements are enforced by peer-pressure or third-parties, if exchange is going to quickly move products around a

[2] As explained in chapter 2, Methodological Individualism is an application of Reductionism to the Social Sciences, where the component to which social phenomena are broken down is individual agents.

society then there need to be as few exchange agreements as possible separating any two individuals. This requirement is obviously equivalent to the small worlds property. In order for agents to substitute away from clustering when third party enforcement becomes preferable, there needs to be some cost to that clustering. It is establishing that cost in the form of a reduced small worlds property, that is the objective of this exercise.

My finding is, however, negative: although strategic behaviour with global information about the network can effectively select a structure with short average path lengths, realistically local information is not enough: when only local information is available, in my model, the network is too unstable to consistently outweigh a pressure towards clustering (see section 5.5).

There exist already several strategic models producing a combination of small worlds and high clustering (see section 5.3), but these all rely on agents each having complete knowledge of the structure of the network; a patently unrealistic assumption. For various reasons, economics has previously employed methodologies that reject the need for observably *realistic* assumptions, most notably in the *As-If* methodology of Friedman. But, as chapter 3 explains, this begs two questions: why restrict what are then effectively holistic models to a structure of "strategic agents", when these do not correspond to the agents that actually make up society? And, when there is often as great a shortage of reliable information on the macroscopic system of interest, as on the microscopic components, why deny oneself the use of what microscopic evidence one has by neglecting its relationship to the macro? To properly supplement weak macroscopic data with empirical evidence from microscopic components, and to avoid the vagueries of abduction, I argued that it was desirable to ensure abstraction only ever removes properties of components that do not affect their aggregate behaviour —negligibility assumptions in the popular typology of Musgrave (1981). In this case a strategic model must not require agents to have global information, unless this assumption can be relaxed with trivial change to the model's behaviour. This chapter therefore extends the

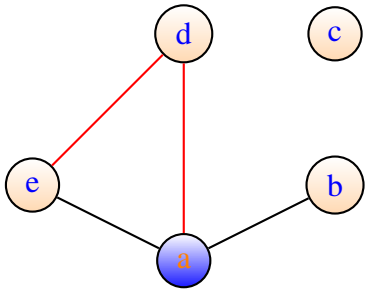


Figure 5.1: Example network

question of whether strategic behaviour can achieve the small-worlds property with only realistically local information.

The chapter is organised as follows: section 5.2 provides a lexicon for network science, reviews the documented evidence for the small worlds property, and details holistic models producing the property; while section 5.3 reviews existing Methodological Individualist models that generate the small worlds property; section 5.4 gives a technical explanation of my approach here; while sections 5.5 describes the model's simulated dynamic and steady state behaviour; finally, section 5.6 provides a summary of my findings.

5.2 The small worlds property

5.2.1 Terminology and notation

Let us refresh the mathematical lexicon and relevant literature for networks, first given in chapter 3, all the while using figure 5.1 as a visual reference:

Node Those objects which are linked within a network — this could be an agent, a firm, or some more abstract object in some uses: $i \in N$. For example a in figure 5.1 is a node, as are b, c, etc.

Edge; Link A connection between two nodes, denoting a relationship of some kind between them: $ij \in g \subseteq \{(k,l) : l,k \in N\}$. For example, the line connecting a, and b, in figure 5.1 represents an edge.

Graph; Network A collection of Nodes and Edges: (N, g). The entirety of figure 5.1 is an example of a graph.

Path A sequence of edges between two nodes, i and j say, including other nodes at most once: $(i_1 i_2, ...i_{I-1} i_I)$ s.t. $i_k i_{k+1} \in g,\ i_1 = i,\ i_I = j$ and $i_k \neq i_l\ \forall\ k,l \in N$. For instance, (ad, de) is the path shown in red in figure 5.1.

Neighbourhood (immediate) The set of nodes which share an edge from g with a particular subject node, i: $N_i = \{j \in N : ij \in g\}$. In figure 5.1, the neighbourhood of a would be $\{b, d, e\}$.

Degree The number of other nodes to which a particular subject node, i, is connected; the cardinality of that node's immediate neighbourhood: $d_i = \mid N_i \mid$. For instance, the degree of a is 3.

Neighbour A node which shares an edge from g with a particular subject node, i: $j \in N_i$. So b is just one neighbour of a.

Neighbourhood The set of nodes that can be reached by crossing no more than a certain number, d , of edges of g, $i : N_i^d = \{j \in N : \exists\, p = (ii_2, ...i_{I-1}j)\ , |p| \leq d\}$

Component A subset of nodes between every pair of which there is a path, but from no element of which there is a path to any other node: N_i^∞. So, in figure 5.1, $\{a, b, d, e\}$ is one component, and $\{c\}$ is the other.

Tree A network or component in which there is only one path between any two nodes. So the graph in figure 5.1 is not a tree.

Clique A sub-network in which every possible link is present: $N' \in N : \forall i,j \in N'\ ij \in g$. So the set $\{a, d, e\}$ is a clique, and in this case a 3-clique or *triad*.

Network Measures

Though the distribution of nodes' degrees alone captures useful features of a network, it fails to differentiate some features typical of observed networks. This motivates some other popular measures of network topology:

Clustering roughly captures the probability with which neighbours share a common neighbour, a kind of density of the network's edges. Within this loose definition there are several more specific measures employed in practice. Local clustering will be considered here, and by extension average clustering will be used to describe the global properties.

$$Cl(g) = \frac{\sum_{ij,ik \in g} \mathbf{1}(jk \in g)}{|ij, ik \in g|}$$

Diameter is the largest, among all pairs of nodes, of shortest paths between two nodes. When something is spreading through a network, the diameter gives an idea of how long it will take to reach all nodes.

Shortest average path length is a diameter-*like* measure of typical distances, that is more robust to outliers. Typically it is taken over the largest component of a network.

5.2.2 The holistic literature on small worlds

Evidence of small-worlds and clustering in social networks (and other networks) is given in many sources, two prominent such are Watts and Strogatz (1998) and Jackson (2010).

Watts and Strogatz's (1998) seminal work, on the small worlds-clustering paradox, contrasts the conventional random networks of Erdos and Renyi with regular lattices: the former (with relatively small degree) have the small worlds property but trivial clustering; the latter have high clustering, but average path lengths that grow algebraically

with network size. Instead Watts and Strogatz propose a model of network formation with both properties. Starting out with a lattice (say, a ring of nodes each connected to two nearest neighbours on each side) existing edges are *rewired* to connect two randomly selected nodes. The effect is to bring the best of both topologies, with the high clustering of the original lattice and the logarithmic growth in path length of random connections.

In Jackson and Rogers (2005a) an alternative mechanism is presented in the form of two different modes of link selection —*search* in their words: nodes are selected sequentially; they are then given links to other nodes randomly selected from those with some links already; finally, the node is given some links to nodes neighbouring those they were just linked to. Among other desirable properties, this means that the complete randomness of the second step leads to small worlds, while the third step naturally leads to clustering. It is a slight refinement of this model that I will use to provide a controlled pressure towards clustering.

5.3 Existing models of strategic network formation

5.3.1 Existing models

The islands connections model of Jackson and Rogers (2005b) is a highly abstracted static equilibrium model that recreates the small worlds property. This model assumes agents receive utility from being linked to others, that decays exponentially with the length of the shortest path between them, $\ell(i,j)$. That is, an agent receives utility δ from an immediate neighbour, δ^2 from a neighbour of a neighbour, and $\delta^{\ell(i,j)}$ from some agent with a minimum path of $\ell(i,j)$ between them. For the sake of tractability a maximum path length that can provide utility is set as D. With this distance based utility in place, links are given a two-tier cost structure. This is explained intuitively as it costing, c to link with an agent on the same *island*, I_i, and C to link with agents on

other *islands*, I_{-i}, with $0 < c < C$. This gives each agent a payoff of,

$$u_i(g) = \sum_{j \neq i : \ell(i,j) \leq D} \delta^{\ell(i,j)} - \sum_{j:ij \in g} (c\mathbf{1}(j \in I_i) + C\mathbf{1}(j \in I_{-i}))$$

It can then be shown that *stable* networks must involve every agent on each island being directly connected, that diameter is no greater than $D + 1$, and that clustering is tightly bounded. However, no dynamics are specified for this framework, so it is unclear whether the equilibrium represents an attractive point. Moreover, agents' consent to links in equilibrium assumes that they have information on distant topology. Both these elements leave in question the status of the model as Methodological Individualist.

The enforcement based model of Jackson et al. (2012) is more sophisticated, with peer pressure introduced as the explicit value of common neighbours, but it suffers from similar problems of dependence on full information.

The network formation process has two stages each time period: first, agents choose which links they would like to *retain* from the previous period; second, a favour is asked at random, by node i of node j, and if it is not performed then the link ij is removed. As with many repeated game scenarios, one equilibrium is the *grim trigger* strategy wherein all agents maintain links and perform favours until one defects, and then all agents sever all links. This is just one of infinitely many possible sub-game perfect strategies, and so the authors introduce an equilibrium refinement concept of *renegotiation–proof–ness*. This makes strategies robust to reconsideration of the grim trigger after defection, by requiring in any subgame that continuing the *renegotiation proof* strategy (and the network it produces) not be Pareto dominated by some other *renegotiation proof* strategy (and the resulting network).

The authors characterise the actual set of renegotiation proof networks recursively, for a particular cost structure. They then further refine their equilibrium concept by looking for *robust* networks, in which the network arrived at after the grim trigger is only changed locally — that is, networks in which there are no cascades of broken

relationships after a single defection. They find that this promotes a unique network structure of complete subnetworks linked in a tree-like structure[3]. These *social quilts* have the high clustering we look for in imitations of the real world phenomena, and a tree-like structure outside those clusters would ensure a logarithmic growth in shortest average path length.

It should be noted that the tractability of the model is dependent on a growth process that involves only deletion of links. The network formation mechanism is therefore not truly dynamic. Moreover, in order for agents to judge whether a link they are offered will be robust, they must again have information of their entire component's topology. As almost any link, added randomly to a quilt, would create a cycle of illicit length, evolutionary pressures cannot be relied upon either to eliminate networks without quilt-like structure — or networks would be continually destroyed by cascades.

Vega-Redondo's enforcement model (2006) does, in contrast, consider dynamic behaviour. It examines first static network equilibria, and then a dynamic formation process, under shifting incentives, by both simulation and mean field approximation[4]. The model is one of perfectly rational agents, for whom a common edge means competing in an infinitely repeated prisoner's dilemma — representing some collective action problem — but also a source of information on the conformity of every agent in the component — though news of defection from the common norm strategy is transferred by only one edge each period. This information transfer service increases conformity of partners to the collaborative strategy, and therefore serves as *collateral* against defection. In its dynamic form, the network grows in a way that borrows from both holistic models described above: at each time period each agent is randomly given the opportunity to form a link with another node within their component of the network or to form a link with any node in the network. Whether or not a link is formed is then determined by whether it offers a Pareto improvement to both agents. With no

[3] With no cycles as large as the complete subnetworks.

[4] A mean field approximation makes a stochastic model deterministic by represents stochastic inputs with their mean value.

variation in the idiosyncratic payoffs to the prisoners' dilemma games, the network almost surely converges to the complete network. However, when the payoffs for each link are redrawn each period, with some incremental probability, ϵ, any equilibrium network arrived at eventually ceases to be an equilibrium and so the system evolves away from it. The process does not arrive at a fixed configuration, but is ergodic and has an unconditional long run distribution (invariant measure), that can be found by simulation.

It is found that, in this framework, increased volatility (ϵ) leads to: lower network density, through more depreciated *social capital* (collateral) each period; more cohesion, that is, lower diameter of components, making what collateral there is more robust to collapsing relationships; a smaller largest component, again representing the instability of less dense social capital; lower average payoff, because with less certain value to relationships it pays to defect more often and so the worst case equilibrium of the prisoners' dilemma is realised more easily.

Despite the more complete treatment, the agents' strategic considerations still rely on having full knowledge of how information will propagate through the network, and hence global topology.

The locality of information is missing from all these treatments, and appears to be an important gap in the literature; I will attempt to address it, starting in the next section.

5.4 The Model

5.4.1 Trade

In order for realistically local strategic behaviour to drive a small diameter, which is a global network property, the relevant global topological information must be available to the agent in some local signal. Here that signal is each agent's vector of goods held from the last period; as it is influenced by that of their neighbours, and hence indirectly

by neighbours of neighbours, etc.. As the good could represent anything of value passed during an interaction between individuals, exchange of goods encompasses all interfaces between strategic behaviour and topology.

In Acemoglu et al. (2012) broadly similar production influences are carried to neighbours instantaneously. Here, however, it is felt that the transfer of goods between neighbours are more realistically conceived as on a similar timescale to production. Hence, a dynamic law of motion is used for the transfer of goods through the network.

Basic trade notation consists of:

x_{ij}^{kt} the amount of good k transferred from agent i to agent j, at time t;

y_i^{kt} the amount of good k produced by agent i at time t.

The type of the good will have implications for the law of motion. If it is rivalrous (e.g. any physical good), then the total value passed to neighbours cannot exceed the value received or produced in the previous period:

$$\sum_{j \in N_i^{t+1}} x_{ij}^{kt+1} \leq y_i^{kt+1} + \sum_{j \in N_i^t} x_{ji}^{kt}$$

However, if the good is non-rivalrous (e.g. information) then it may be transferred in quantity equal to that received. Conversely, it cannot be received more than once. This means that only the shortest path to the source will have value — except where topology has recently changed and earlier bits have not been received by that shortest path. Where there is any depreciation of the good as it passes through the network, the shortest path will be captured by the maximum quantity of the good among transfers:

$$x_{ij}^{kt+1} \leq \max\{y_i^{kt+1}, x_{li}^{kt} : l \in N_i^t\}$$

The literature on social networks largely treats the value transmitted through a network as being information. Because of this, and because it is more tractable, I will focus on non-rivalrous goods in this chapter.

The Law of Motion could take many conceivable forms, depending on how agents choose to allocate the good among their neighbours. Indeed, in reality one might expect strategic bargaining, and some attempts at manipulation. For simplicity, I will assume non-rivalrous goods are offered to all neighbours without exception. Whether this approximating abstraction is trivial is a matter for further enquiry, but there seems no intuitive reason that it would affect the system's behaviour. It would also be reasonable to conceive of a good like information degrading in quality as it is passed through the network (the game Chinese Whispers relies on this property); hence, a depreciation multiplier, δ, is included. The law of motion for a non-rivalrous good then becomes,

$$x_{ij}^{kt+1} = \delta \max\{y_i^{kt+1}, x_{li}^{kt} : l \in N_i^t\}$$

5.4.2 The Agents

The stated aim of this chapter is to establish the limits of strategic effects on small-worlds topology under realistically local information. Clearly this requires a precise and intuitive model of individual agent behaviour.

Agent specific goods production will be assumed, meaning that each agent produces a unique good; in order to encourage short paths between every pair of nodes there needs to be value for each agent, of paths to each other agent, that cannot be perfectly substituted. Letting this production be of a uniform amount, π, across agents,

$$= \pi \quad \text{if } i = k \tag{5.1}$$

$$y_i^{kt} \tag{5.2}$$

$$= 0 \quad \text{if } i \neq k \tag{5.3}$$

A bound on the number of neighbours that an agent can have is introduced, so as to prevent the network simply building up links until it becomes complete. Although a complete network would have unit diameter, it is quite patently unrealistic for every person in a society to interact directly with every other. I will designate this bound as M.

Linear utility in the various goods, is a simplifying abstraction from the more conventional and intuitive convex case. With utility only consumed from a single source, however, there can be no competition between incentives for single and multiple paths to other agents. Intuitively therefore, convexity should not change the basic incentives to have minimal path lengths to as many other agents as possible — only the value of many long paths relative to a few short paths. Simulation of the system with exponential utility confirms this intuition, as there is no qualitative change in behaviour — however I do not reproduce those results here. Hence, agents' utility from their relationships will be captured by the function,

$$u_i^t = \frac{1}{|N|} \sum_{k \in N} z_i^{kt}$$

I also assume *quasi-myopic decision making*: spacial locality of information is an assumed property of the physical world, in all but the quantum realm, so limiting strategy to local space should not be controversial to readers. There is, however, experimental and simulation evidence suggesting that real humans' decision making is likely also bounded in its temporal scope: repeated game equilibrium refinements like backward induction are not supported by experimental evidence (Johnson et al., 2002); *time inconsistency* (hyperbolicity) is often observed in the time preferences of humans in experimental settings (Strotz, 1955-56); meanwhile, simple trial and error algorithms do not distinguish well between subgame perfect and other Nash equilibria (Samuelson, 1994). For this reason, I choose not to model agents' decision making as taking account of all future transactions.

Because the goods are not spoilable, agents will carry their stocks of goods from an old topology to their position in a new topology. The value of these instantaneous transfers are an obvious bias in the short run, and so nor do I treat decision making simply as myopic. Instead, each agent will approximate the goods bundle to be expected under the new topology, by the goods bundle expected one period after the topology change. That is, they subtract any goods the new neighbour would have received from the neighbour they abandon, and (trivially) add the goods bundle they would pass to the new neighbour.

The search mechanism, for new neighbours, will serve two purposes: it will allow agents the opportunity to choose between different neighbours, thereby bringing their strategic behaviour to bear on network topology; it will also reproduce a constant pressure towards high clustering, in opposition to any strategic pressure towards short path lengths. Toward this purpose, I employ a mechanism with both conventional global random search and *local* search encouraging clustering. This mechanism is closest to that of Vega-Redondo (2006). Each period, the search illustrated in figure 5.2 consists of:

i: A subject agent is chosen uniformly at random from the population;

iia: EITHER, with probability p, a second agent is then chosen uniformly from those that are not currently neighbour to the first;

iib: OR, with probability $1 - p$, a neighbour of the agent is chosen uniformly at random, and a neighbour of theirs chosen as a potential new neighbour — if any neighbour is not also a neighbour to the subject, otherwise a different existing neighbour is chosen uniformly from the remaining neighbours;

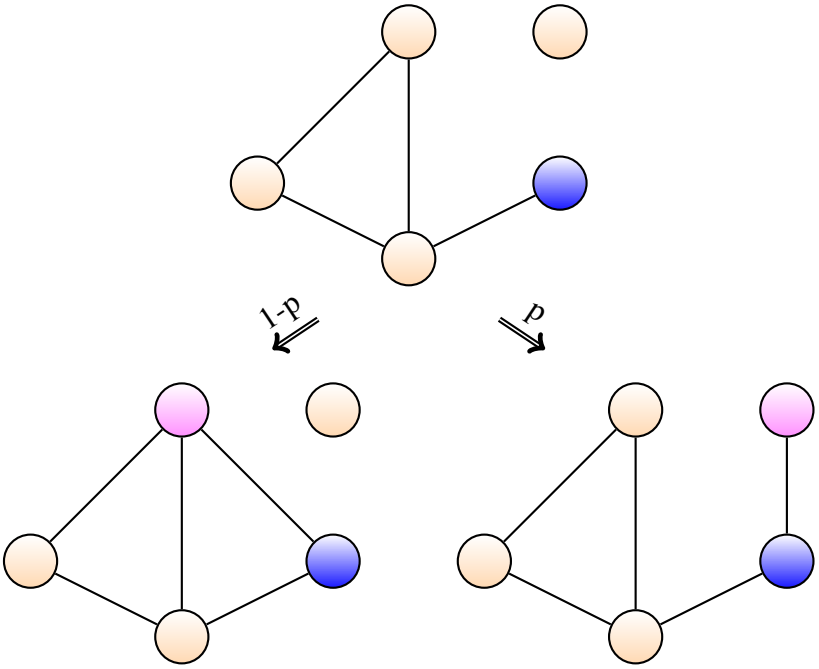


Figure 5.2: The neighbour search process

5.4.3 The sequence of events

In light of the above modelling framework, the following series of decisions and events will take place within each time period:

1: Exchange Transfers are made between all pairs of neighbours, and their subsequent stocks of the goods are updated;

2: Search As detailed above, an agent is chosen uniformly at random, and a new potential neighbour is found for them according to the search algorithm;

3: Decisions Each of the newly selected pair of agents observes the other's neighbourhood. They then judge what maximal combination, of their existing neighbours and the new neighbour, would maximise their utility. If the other agent already has a maximal number of relationships, it is assumed in strategising that they would retain the most valuable combination of size one less than the maximum;

5.5 Behaviour of the model

Even where agents are otherwise perfectly homogeneous, the importance of their position within a network can lead to extremely high dimensionality when describing their collective behaviour — unless, say, the topology were itself somehow forced to be regular. For this reason, it is rarely possible to capture the precise dynamics of a network system, and we must settle instead for either: approximations, such as mean-field analysis; or a precise account of behaviour for only some finite set of parameter values, in the form of simulated data. In order to understand the dynamic behaviour of this model, I focus here on the latter option; all the while conscious that some behaviours may remain unobserved in the gaps between tested parameters.

Initial conditions are likely to be important for the models, as they can easily be seen to be non-ergodic [5]. Nevertheless, it may be assumed that any social network starts with some disconnected state, as the members will all have finite lifespans. For this reason, the initial condition for each simulation will be a network with no relationships.

Stability will be important for the emergence of the small worlds property in a network, under a competing pressure for clustering: it necessarily takes time to build up paths between many pairs of nodes. So, even if links are initially selected in such a way that path lengths should be minimised, if there is not also a pressure to maintain these links then short paths will not build up.

5.5.1 Steady state behaviour

My principal interest is in the long run behaviour of topological properties. Without ergodicity the system may spend time in states that will not then be revisited. These states will be of trivial importance in the long run, and so they should be disregarded in an investigation of the topological properties. I therefore include a *burn in* period

[5] In order for links to be severed, another link must be created. Hence, the system can never return to a fully disconnected state.

in the simulations, before taking the mean of topological properties over subsequent iterations to capture long run properties. The sets of states other than these transient ones may be regarded as *steady states*. As the system is stochastic, different paths may be taken, and different steady states entered on different instantiations of the model. For this reason the behaviour of the models must also be considered over multiple repetitions starting from the initial conditions. For figure 5.3 I average across these repetitions in order to gain a description of general behaviour. This specific example is a network of *100 agents*, each able to sustain at most *4 relationships*. Values are averages over *20 simulations*. Each simulation consisted of *50,000 iterations*, with the first 10,000 abandoned to allow a steady state to be approached. Means of each metric were then taken over the following 40,000 iterations, before being averaged.

Full Information

The topology under perfect information is a useful benchmark for the local information case. For the non-rivalrous good, where only the shortest path a good can take to the agent is relevant, full information involves knowing these path lengths for any given configuration of neighbours. The first step of the process above then becomes trivial. Meanwhile, in the third step agents now base their decisions on the known path lengths rather than the topological-information contained in neighbours' inventories.

Observation 5.5.1 Regardless of the pressure towards clustering, $1-p$, strategic agents acting on full information of path lengths will maintain a connected network with path lengths comparable to a network built by random global search.

Simulations support this observation, under a broad range of parameters; figure 5.3 shows behaviour for the illustrative example of 100 agents, each capable of maintaining relationships to four other agents. It can be seen that networks generated by these *hyperopic* agents have extremely stable properties as the probability of search being local varies. Notably, the average path length remains that of a fully random network,

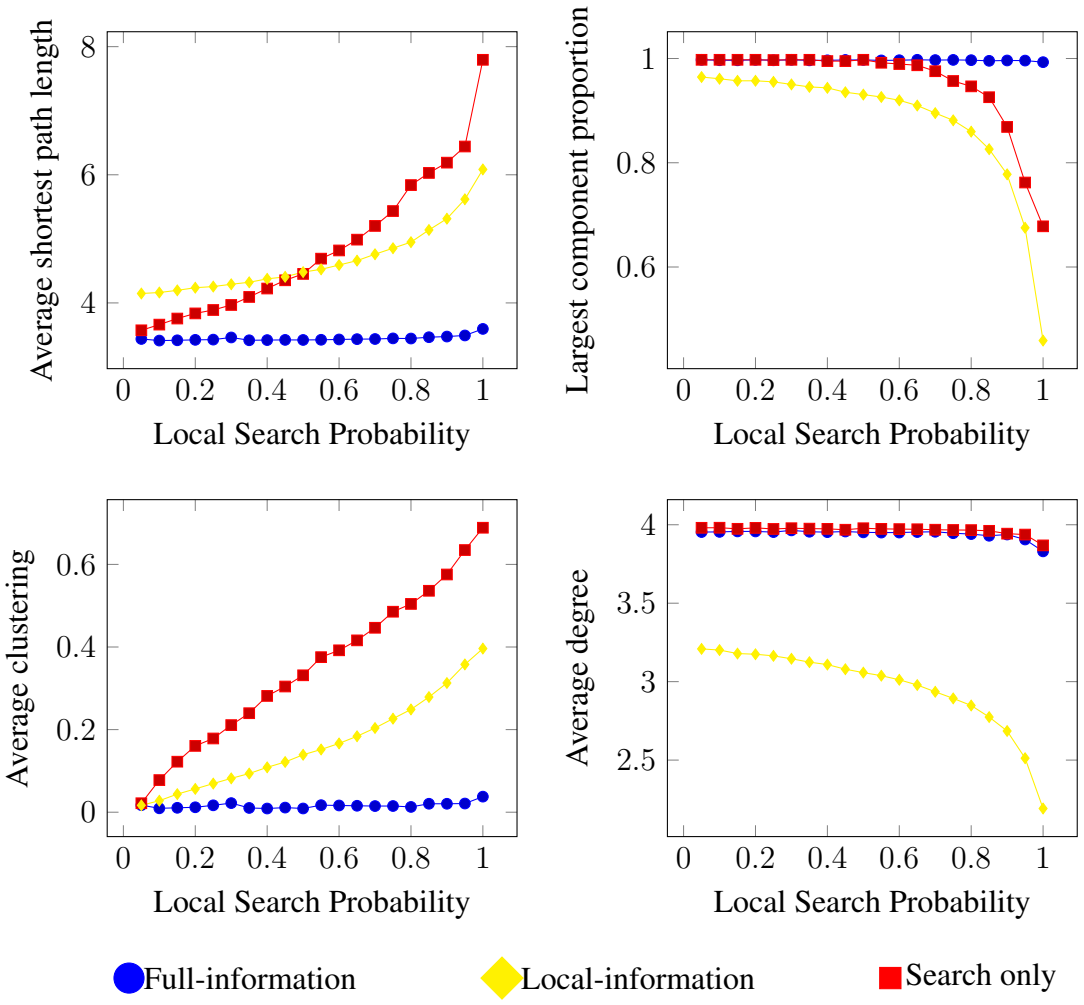


Figure 5.3: Steady state values of key network metrics for varying probabilities of local search.

while the search mechanism alone diverges towards the path length we would expect of a geometric relationship —given the component sizes. While the graphs generated by search are connected for values of p above a certain point, they become disconnected when local search becomes too common. The strategic agents, however, maintain a connected graph for all values of the search parameter.

Local information

Non-rivalrous goods can capture path lengths in the form of the least depreciated supply among neighbours; in the important example of information, we can imagine recognising the expertise of a particular neighbour on a given topic by the additional detail of their account. Is there enough topological information contained in this measure? The answer is no, arguably because goods we have previously passed to neighbours mask whether they could also have received those same goods (with the same depreciation) via another path of the same length. More formally,

Observation 5.5.1 As the pressure towards clustering, $1-p$, increases, strategic agents acting on only local information fail to compensate for pressures towards clustering: the average path length within the generated network diverges from the logarithmic case.

Simulations support this observation, under a broad range of parameters; figure 5.3 shows behaviour for the illustrative example of 100 agents, each capable of maintaining relationships to four other agents. In contrast to the full information case, it is apparent that average path lengths grow as p falls. This is despite the fact that the component over which these paths are being measured shrinks. The shrinkage is also notable because it exceeds that of the purely random generating mechanism. A clue to both differences in behaviour comes in the average degrees of the agents in the various models. It is found that agents maintain full degree in both the random and full information models, for all values of p; meanwhile, in the local information case agents fail to maintain full degree, and this worsens as p falls. The mechanisms' design applies constant pressure towards agents achieving full degree (of four in the example), and so this failure can only occur because of some instability causing a large set of relationships to be broken even after being chosen.

An intuitive justification for this difference in behaviour, from the full information case, comes with the realisation that a good passed on to a neighbour by the agent

masks other sources of the good to that neighbour. This prevents the agent from compensating for goods in a neighbour's possession that have come from the agent themself. Without compensating for these *reflected* goods, even when a unique path exists through a neighbour to some good's source the agent will perceive that there is another path via each of their other neighbours. These *reflected* paths will each have a length only three steps greater than the true unique path: a step to the agent from the neighbour hosting the unique path; a step from the agent to the other neighbour; and a step back from that other neighbour to the agent. For this reason, agents will willingly allow unique paths between many of the network's nodes to be destroyed, thinking that there are alternatives. In our information example, once we have already told a piece of news to a friend, we cannot then know whether they could also have heard this news from another source —at least, not from the simple fact of them knowing the news.

5.5.2 Dynamic behaviour

In light of finding that the model with local information does not converge to a single state, its dynamic behaviour becomes of interest to its long run properties. This dynamic behaviour is illustrated in figure 5.4 for five simulations with local information (left), and with full information (right). This specific example is a network of *100 agents*, each able to sustain at most *4 relationships*.

Highly unstable behaviour is observed, as guessed, for the model with local information only in figure 5.4: as local search probability, the pressure towards clustering, increases the strategic pressure towards a single component with short path lengths becomes insufficient to overcome it. This is in stark contrast to the model with full information on path lengths, in which convergence is slowed but still assured even under high pressure towards clustering. For the local information case, it might appear that extremely short paths are still sometimes achieved. However, comparison with the component sizes makes it clear that these periods of low average shortest paths coincide with periods where the network has only a small largest component — with low

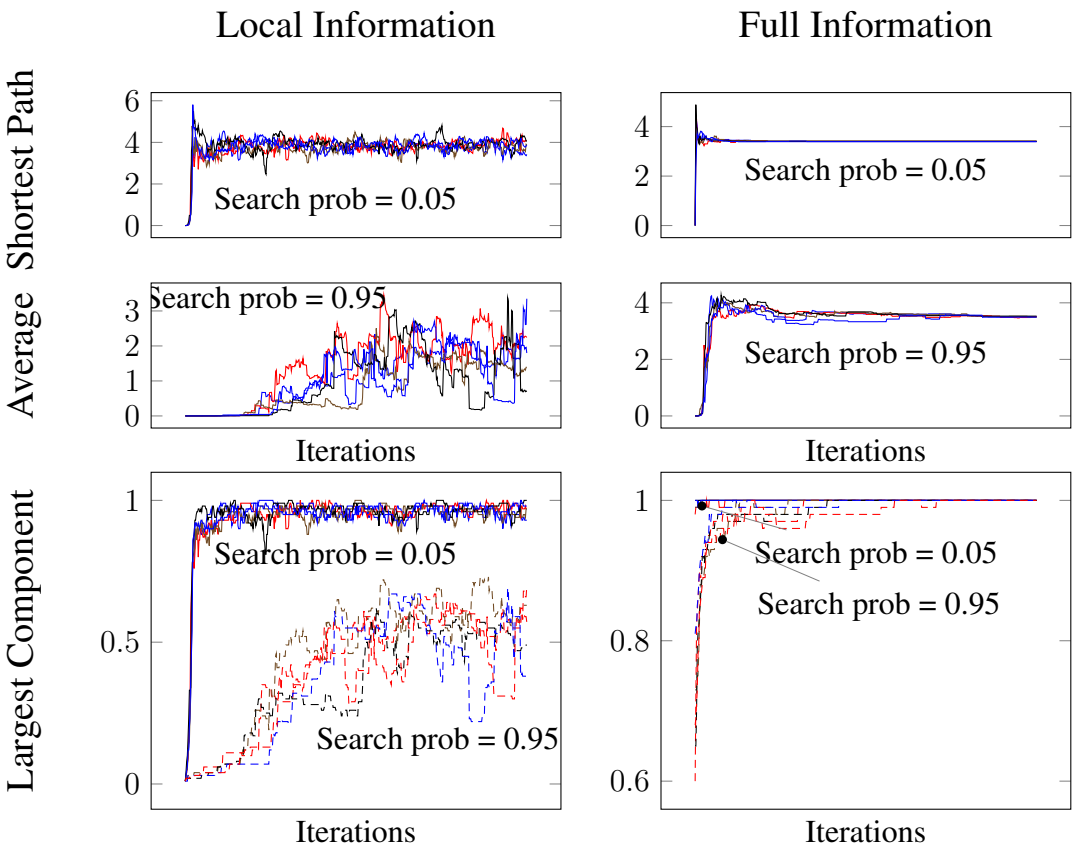


Figure 5.4: Dynamic development of simulations under extremal values of local search probability.

average shortest paths a necessary consequence.

5.6 Conclusions and extensions

5.6.1 This chapter

I have argued that the current literature on strategically formed networks neglects realistic limits on the locality of information, with regard to generally observed small worlds property. Further, the case was made that small worlds could not arise without some strategic mechanism, that can work on local information only, in the presence of a constant pressure towards the widely observed stylised fact of high clustering. Hav-

ing constructed a natural strategic model of network formation, simulations showed that local information is not sufficient to produce the topological features observed in reality. This observation should be taken very seriously by any modeller choosing a mechanism for clustering that provides unchanging incentives, as can be seen in such staples as the islands-connections model of Jackson and Rogers (2005b). It does not, however, rule out the small worlds property emerging in a Methodological Individualist model: given that random search can produce the property when not opposed by pressure towards clustering, it seems likely that this would suffice so long as incentives towards high clustering disappear in the presence of high clustering.

In order for my research program to further its attempt at describing economic growth, I have suggested that peer-pressure based enforcement should be modelled and set in opposition to the hierarchical enforcement modelled in chapter 3. In order for such a model to successfully describe social networks, the problems raised in this chapter must be addressed. I leave this a topic for further research.

5.6.2 This research program

The negative result of this chapter marks a premature conclusion to the Macroeconomics of Social Contracts. But, in truth it should be no more than a hiatus. The methodological arguments of chapter 2 still urge for new approaches to Macroeconomics, besides the logically and epistemologically flawed General Equilibrium based models. Meanwhile, chapters 3 and 4 lay the foundations for a logically superior, and as yet unrefuted alternative model of macroeocnomic fluctuations. As stated above, the result above does not prohibit the intuitive model of macroeconomic growth that I have sketched here and in chapter 1. I hope that I have impressed on the reader a need to help me continue this work. If not that, then I hope that I have at least shown that there is nothing to fear in leaving the well trodden path of Orthodox Macroeconomics, and that the vast virgin wilderness beyond is ripe with opportunity.

Bibliography

Acemoglu, D., Carvalho, V. M., Ozdaglar, A., and Tahbaz?Salehi, A. The network origins of aggregate fluctuations. *Econometrica*, 80(5):1977–2016, 09 2012. URL http://ideas.repec.org/a/ecm/emetrp/v80y2012i5p1977-2016.html.

Boccara, N. *Modeling Complex Systems*. Graduate Texts in Contemporary Physics. Springer, 2004. ISBN 9780387404622.

Coleman, J. S. Social capital in the creation of human capital. *American journal of sociology*, pages S95–S120, 1988.

Jackson, M. *Social and Economic Networks*. Princeton University Press, 2010. ISBN 9780691148205.

Jackson, M. O. and Rogers, B. W. The economics of small worlds. *Journal of the European Economic Association*, 3(2-3):617–627, 04/05 2005a.

Jackson, M. O. and Rogers, B. W. The economics of small worlds. *Journal of the European Economic Association*, 3(2-3):617–627, 04/05 2005b.

Jackson, M. O., Rodriguez-Barraquer, T., and Tan, X. Social capital and social quilts: Network patterns of favor exchange. *American Economic Review*, 102(5):1857–97, August 2012. URL http://ideas.repec.org/a/aea/aecrev/v102y2012i5p1857-97.html.

Johnson, E. J., Camerer, C. F., Sen, S., and Rymon, T. Detecting failures of backward induction: Monitoring information search in sequential bargaining. *J. Economic Theory*, 104(1):16–47, 2002.

Musgrave, A. "unreal assumptions" in economic theory: The f-twist untwisted. *Kyklos*, 34(3):377–387, 1981. ISSN 1467-6435.

Samuelson, L. *Does evolution eliminate dominated strategies?* The Frontiers of Game Theory. MIT Press, Cambridge, MA, 1994.

Strotz, R. Myopia and inconsistency in dynamic utility maximization. *Review of Economic Studies*, 23(3):165–180, 1955-56.

Travers, J. and Milgram, S. An experimental study of the small world problem. *Sociometry*, 32(4):p425 – 443, 1969. ISSN 00380431.

Vega-Redondo, F. Building up social capital in a changing world. *Journal of Economic Dynamics and Control*, 30(11):2305–2338, November 2006.

Watts, D. J. and Strogatz, S. H. Collective dynamics of small-world networks. *Nature*, 393(6684):440–442, June 1998.

5.A Simulation Codes

5.A.1 Standard Exchange Network Model

Agent.java

```
/*
 Copyright 2011 by Tom Wilkinson and Cardiff University
 Licensed under the Academic Free License version 3.0
 See the file "LICENSE" for more information
*/
package anarchytrade;
import util.*;
import java.text.*;
```

```
/* This is an agent object that populates a social network. It
    assesses
* relationships according to local information, and looks
   ahead only
* one step.
*/
public class Agent {
   DecimalFormat df = new DecimalFormat("#.###");

   // fixed
   final int index;
   final int numAgents;
   final int maxRelationships;
   final double depreciation;
   final double innovation;
   final double invNumAgents;
   final double imitation;

   // variable
   double[] diversity;
   double diversityMeasure;
   double[] estimatedDiversity;
   double estimatedDivMeasure;
   double estimatedUtility;
   int guessRejectedInd;
   int product;

   Bag neighbours;
   Bag oldNeighbours;
   Bag component;

   int firstNeighbourIndex;
   double firstNeighbourDiversity;

   public Agent(int numAgents, int index, int maxRelationships
      ,
         double depreciation, double innovation){
      this.index = index;
      this.numAgents = numAgents;
      this.maxRelationships = maxRelationships;
      this.depreciation = depreciation;
      this.innovation = innovation;

      invNumAgents = 1.0/(double)numAgents;
      imitation = 1.0 - innovation;
```

```
        diversity = new double[numAgents];
        diversity[index] = 1.0;

        neighbours = new Bag();
        oldNeighbours = new Bag();
        component = new Bag();
    }

    /* With all the agents' ownProducts established we now need
        to take
     * transfers into account
     */
    public double[] receiveTransfers() {
        double[] updatedDiversity = new double[numAgents];

        int numNeighbours = neighbours.numObjs;

        /* We'll work out the quantity, that the agent has, of
           each property */
        for(int property=0; property<numAgents; property++) {
            double newProperty = 0.0;
            if(property==index)
                newProperty = innovation;
            else {
                for(int nbrIndex=0; nbrIndex<numNeighbours;
                   nbrIndex++){
                    Agent neighbour = (Agent)neighbours.objs[
                       nbrIndex];
                    double nbrProperty = neighbour.diversity[
                       property];
                    if(nbrProperty>newProperty)
                        newProperty = nbrProperty;
                }
                newProperty *= depreciation;
            }
            updatedDiversity[property] = newProperty;
        }
        return updatedDiversity;
    }

    public void removeNeighbours(Bag formerNeighbours) {
        for(int i=0; i<formerNeighbours.numObjs; i++) {
            // we remove the neighbour
            neighbours.remove(formerNeighbours.objs[i]);
        }
    }
```

```
}
```

SocialNetwork.java

```
/*
 Copyright 2011 by Tom Wilkinson and Cardiff University
 Licensed under the Academic Free License version 3.0
 See the file "LICENSE" for more information
*/
package anarchytrade;
import util.*;

/* Anarchy Trade model allows agents to predict the effects of
    adding neighbours
* on those neighbours' diversities!
*/
public class SocialNetwork {
    Bag allNodes;
    MersenneTwisterFast random;

    final boolean localSearch;

    final int numAgents;
    final int maxRelationships;
    final int exchangeRate;
    final double globalSearchProb;

    final double depreciation;
    final double innovation;

    final double invNumAgents;

    Bag newEdges;

    int edgesRemoved;
    boolean statsTaken;
    Bag components;
    Bag largestComponent;
    Bag secondComponent;

    // instantaneous network statistics:
    double averageDegree;
    int[] degreeDistribution;
    double averageClustering;
    double[] clusteringDistribution;
    double clustering;
    double support;
```

```
double[] centrality;
double largestComponentShare;
double secondComponentShare;
double averageShortestPath;
double averageEdgeAge;

Bag parents;

// dynamic network statistics:
double averageProduct;
double averageDiversityMeasure;
int numRelationships;
double completeness;
int relationshipsBroken;

public SocialNetwork(boolean localSearch, int numAgents,
   int maxRelationships,
        int exchangeRate, double globalSearchProb, double
           depreciation,
        double innovation) {
    this.localSearch = localSearch;

    this.numAgents = numAgents;
    this.maxRelationships = maxRelationships;
    this.exchangeRate = exchangeRate;
    this.globalSearchProb = globalSearchProb;
    this.depreciation = depreciation;
    this.innovation = innovation;

    invNumAgents = 1.0/(double)numAgents;

    newEdges = new Bag();

    random = new MersenneTwisterFast();
}

// Transfers are now called in from the various agents, and
    then redistributed
public void exchangeTransfers() {
    double[][] diversities = new double[numAgents][
       numAgents];

    // first, let's call in each of the updated diversity
       arrays
    for(int agentIndex=0; agentIndex<numAgents; agentIndex
       ++) {
```

```
            Agent agent = (Agent)allNodes.objs[agentIndex];
            diversities[agentIndex] = agent.receiveTransfers();
        }

        // now, with all agents' diversities updated, the
           original diversities
        // are no longer needed and can be replaced
        for(int agentIndex=0; agentIndex<numAgents; agentIndex
           ++) {
            Agent agent = (Agent)allNodes.objs[agentIndex];
            agent.diversity = diversities[agentIndex];
        }
    }


    /* After transfers but before choices, this provides new
       neighbours to those
     * agents that can sustain them
     */
    public void neighbourSearch() {
        Bag classifieds = new Bag(allNodes);

        /* local search should happen first, to reduce the
           possibility
         * that there are no viable local search choices left
            in classifieds:
         * - if there is a clique there could still be no local
             search choices */

        Agent agent =
                (Agent)classifieds.objs[random.nextInt(
                   classifieds.numObjs)];

        // we create a new bag to keep track of neighbours we'
           ve visited
        // in local search
        Bag neighbours = new Bag(agent.neighbours);

        // now a specific classifieds bag stopping us selecting
        // inappropriate new links for this particular agent
        Bag tempClassifieds = new Bag(classifieds);
        tempClassifieds.remove(agent);
        tempClassifieds.removeAll(neighbours);
        if(tempClassifieds.isEmpty()) {
            classifieds.remove(agent);
        }
```

```
// with a given probability a new neighbour is found by
    global search
if(random.nextBoolean(globalSearchProb)) {
    int potentialClassified =
            random.nextInt(tempClassifieds.numObjs);
    Agent potentialNeighbour = (Agent)tempClassifieds
            .objs[potentialClassified];
    /* now we add these agents as a pair to the bag of
       new edges
     * to check out */
    newEdges.add(new Agent[]{agent,potentialNeighbour})
        ;

    classifieds.remove(agent);
    classifieds.remove(potentialNeighbour);
}
// otherwise local search is used - may not be
   successful!
else if(localSearch) {
    // first we randomly choose a neighbour from whom
       to pick a
    // common neighbour
    while(!neighbours.isEmpty()) {
        int neighbourIndex = random.nextInt(neighbours.
           numObjs);
        Agent viaNeighbour = (Agent)neighbours.objs[
           neighbourIndex];

        Bag potentials = new Bag(viaNeighbour.
           neighbours);
        potentials.remove(agent);

        // now we randomly search through that agent's
        // neighbours
        while(!potentials.isEmpty()) {
            int potentialIndex = random.nextInt(
               potentials.numObjs);
            Agent potential = (Agent)potentials.objs[
               potentialIndex];

            if(tempClassifieds.contains(potential)) {
                newEdges.add(new Agent[]{agent,
                    potential});

                classifieds.remove(agent);
```

```
                    classifieds.remove(potential);
                    return;
                }
                else potentials.remove(potential);
            }
            neighbours.remove(viaNeighbour);
        }
    }
}

/* Agents evaluate relationships, they are then ranked, and
    this is followed
 * by the actual change to network topology
 */
public void neighbourChoose() {
    Bag[] neighboursToRemove = new Bag[numAgents];
    int numNewEdges = newEdges.numObjs;

    /* When agents are making choices about links, they may
        assume that their
     * neighbours' sets of neighbours will remain unchanged
        (as their decision
     * will have no effect on this)
     * You can, therefore, work out every agent's basic
        transfer value
     * beforehand, and new values need only be calculated
        for the agents
     * party to a potential new link in each combination */
    double[][] transferValues = new double[maxRelationships
        +1][numAgents];
    for(int agentIndex=0; agentIndex<numAgents; agentIndex
        ++) {
        double[] estmtdNbrProperties = new double[numAgents
            ];

        Agent agent = (Agent)allNodes.objs[agentIndex];
        Bag neighbours = agent.neighbours;
        int numNeighbours = neighbours.numObjs;
        double transferValue = 0.0;
        /* We'll work out the quantity, that the agent has,
            of each property */
        for(int property=0; property<numAgents; property++)
            {
            /* The agent will inherit, from their
                neighbours, a share of
```

```
             * only the largest value of a particular
                property */
            double newProperty = 0.0;
            if(agent.index==property) {
                newProperty = innovation;
            }
            else {
                for(int nbrIndex=0; nbrIndex<numNeighbours;
                    nbrIndex++) {
                    double nbrProperty = ((Agent)neighbours
                       .objs[nbrIndex])
                            .diversity[property];
                    if(nbrProperty>newProperty) {
                        newProperty = nbrProperty;
                    }
                }
                newProperty *= depreciation;
            }
            transferValue += newProperty;
            estmtdNbrProperties[property] = newProperty;
        }
        transferValue *= invNumAgents;
        transferValues[0][agentIndex] = transferValue;
        agent.estimatedDiversity = estmtdNbrProperties;
        agent.estimatedDivMeasure = 0.0;
        agent.guessRejectedInd = -1;
        agent.estimatedUtility = 0.0;
    }


    newEdgeLoop: for(int newEdgeIndex=0; newEdgeIndex<
       numNewEdges; newEdgeIndex++) {
        Agent[] agents = (Agent[])newEdges.objs[
           newEdgeIndex];
        Agent[] rejects = new Agent[2];

        /* The utility will be derived from a number of
           transfers
         * corresponding to exchangeRate. The prediction
            for utility will
         * therefore take into account
         * - a single transfer based on the new neighbour's
            former neighbourhood
         * - a series of transfers based on their would-be
           neighbourhood
```

```
     *   (these could be discounted exponentially, or
         hyperbolically)
     *
     * For the sake of their new neighbour predicting
         the
     * indirect topological effects of initiating a
         relationship,
     * the following loop will assess which of each
         neighbour's existing
     * relationships is the least valuable to them
     * agents will then assume that their neighbour
         abandons their least
     * valuable relationship in judging their value */
    for(int nodeIndex=0; nodeIndex<2; nodeIndex++) {
        Agent agent = agents[nodeIndex];

        Bag neighbours = new Bag(agent.neighbours);

        int numNeighbours = neighbours.numObjs;
        int maxNeighbours = maxRelationships-1;
        /* if the agent can sustain all links, then
           they will - it is
         * assumed that all links will provide positive
             value or defection
         * will just lead to equal value */
        if(numNeighbours<maxRelationships) continue;
        /* if there are too many neighbours to sustain,
             then we'll assess
         * the value of all combinations (size of
             maxRelationships-1) */
        Bag combination = new Bag();
        Bag combinations = new Bag();
        getCombinations(neighbours, -1, maxNeighbours,
           combination,
                combinations);
        int numCombinations = combinations.size();
        Bag bestCombination = new Bag();
        double bestCombinationValue = 0.0;

        /* now to explore all the possible combinations
             of neighbours */
        for(int combinationIndex=0;
                combinationIndex<numCombinations;
                combinationIndex++) {
            combination = (Bag)combinations.objs[
               combinationIndex];
```

```
/* now an implementation with non-separable
    utility:
 * - as the effects of the edge's other
    party are not being
 *   considered, there is no need to
    consider different
 *   different initial and subsequent
    neighbourhoods */
double combinationValue = 0.0;

/* Now to add the transfers for each of the
    neighbours in
 * this combination */
for(int property=0; property<numAgents;
    property++) {
    double combProperty = 0.0;

    /* here we get the agent's retained
       property, as would
     * have been received the previous
       period given their
     * neighbourhood in this combination */
    if(property==agent.index) {
        combProperty += agent.innovation;
    }
    for(int nbrIndx=0; nbrIndx<
       maxNeighbours; nbrIndx++) {
        Agent neighbour = (Agent)
           combination.objs[nbrIndx];

        double nbrProperty = neighbour.
           diversity[property];
        if(nbrProperty>combProperty){
            combProperty = nbrProperty;
        }
    }
    combProperty *= depreciation;


    /* here we add each neighbour's
       estimated property */
    for(int nbrCombIndex=0;
            nbrCombIndex<maxNeighbours;
            nbrCombIndex++) {
```

```
                    Agent neighbour = (Agent)
                        combination.objs[nbrCombIndex];
                    combProperty += neighbour.
                        estimatedDiversity[property];
                }

                combinationValue += combProperty;
            }
            combinationValue *= invNumAgents;

            /* If this combination generates a higher
                utility than the
             * existing best, then this will replace it
                */
            if(combinationValue>bestCombinationValue) {
                bestCombination = combination;
                bestCombinationValue = combinationValue
                    ;
            }
        }
        /* Having found the best combination of
            neighbours for this agent,
         * we find the neighbour that it does not
            contain */
        for(int nbrIndex=0; nbrIndex<neighbours.numObjs
            ; nbrIndex++) {
            if(!bestCombination.contains(neighbours.
                objs[nbrIndex])) {
                rejects[nodeIndex] = (Agent)neighbours.
                    objs[nbrIndex];
                agent.guessRejectedInd = rejects[
                    nodeIndex].index;
            }
        }
    }

    /* the following loop will assess the options for
        each agent,
     * continuing the thisEdge loop if either
        dissapproves, but allowing
     * the link to be added if both approve */
    Bag[] bestCombination = new Bag[2];
    for(int nodeIndex=0; nodeIndex<2; nodeIndex++) {
        Agent agent = agents[nodeIndex];
        Agent newNeighbour = agents[(nodeIndex+1)%2];
```

```
Bag neighbours = new Bag(agent.neighbours);
neighbours.add(newNeighbour);

int numNeighbours = neighbours.numObjs;
/* if the agent can sustain all links, then
   they will - it is
 * assumed that all links will provide positive
     value or defection
 * will just lead to equal value */
if(numNeighbours<=maxRelationships) continue;

/* If a link is formed, then the transfer value
    of the
 * newNeighbour will be different from that
    produced above.
 * This agent will assume that the new
    neighbour will, if
 * forming a link, abandon the old neighbour
    that didn't feature
 * in their most valuable combination of other
    neighbours */
double newNbrTransferValue = 0.0;
double[] newNbrDiversity = new double[numAgents
   ];
Agent nbrReject = rejects[(nodeIndex+1)%2];
Bag nbrNeighbours = newNeighbour.neighbours;

if(nbrNeighbours.numObjs<maxRelationships) {
    newNbrTransferValue
            = transferValues[0][newNeighbour.
               index];
    newNbrDiversity = newNeighbour.
       estimatedDiversity;
}
else {
    for(int property=0; property<numAgents;
       property++) {
        double combProperty = 0.0;

        if(property==newNeighbour.index) {
            combProperty += newNeighbour.
               innovation;
        }

        double rejectLoopProp = nbrReject.
           diversity[property];
```

```
            rejectLoopProp *= rejectLoopProp;
            for(int nbrIndx=0; nbrIndx<
                maxRelationships; nbrIndx++) {
                Agent neighbour = (Agent)
                    nbrNeighbours.objs[nbrIndx];

                double nbrProperty = 0.0;
                if(neighbour!=nbrReject) {
                    nbrProperty = neighbour.
                        diversity[property];
                }
                if(nbrProperty>combProperty){
                    combProperty = nbrProperty;
                }
            }
            double agentProperty = agent.diversity[
                property];
            if(agentProperty>combProperty)
                combProperty = agentProperty;

            combProperty *= depreciation;

            newNbrTransferValue += combProperty;
            newNbrDiversity[property] =
                combProperty;
        }

        newNbrTransferValue *= invNumAgents;
        newNeighbour.estimatedDivMeasure =
            newNbrTransferValue;
    }

    /* if there are too many neighbours to sustain,
        then we'll assess
     * the value of all combinations (size of
        maxRelationships) */
    Bag combination = new Bag();
    Bag combinations = new Bag();
    getCombinations(neighbours, -1,
        maxRelationships, combination,
            combinations);
    int numCombinations = combinations.size();
    double bestCombinationValue = 0.0;

    /* now to explore all the possible combinations
        of neighbours */
```

```
for(int combinationIndex=0;
        combinationIndex<numCombinations;
        combinationIndex++) {
    combination = (Bag)combinations.objs[
        combinationIndex];
    /* For each combination, we need to find
       the value to the agent
     * of this set of edges:
     * - The neighbourhoods of both the agent
       and the new
     *   neighbour will change, but only after
       the first exchange
     *   HOWEVER WE WILL ASSUME THE FORMER
       NEIGHBOUR's PROPERTIES
     *   THAT WOULD HAVE BEEN CARRIED OVER FOR
       THE FIRST EXCHANGE
     *   WILL BE OFFSET BY A COST TO NEW
       RELATIONSHIPS */

    double combinationValue = 0.0;

    /* Now to add the transfers for each of the
       neighbours in
     * this combination */
    for(int property=0; property<numAgents;
        property++) {
        double combProperty = 0.0;

        /* the agent will consume all of their
           own product that
         * is not transferred;
         * here we get the agent's updated
           property in this
         * combination */
        if(property==agent.index){
            combProperty += agent.innovation;
        }

        for(int nbrIndx=0; nbrIndx<
            maxRelationships; nbrIndx++) {
            Agent neighbour = (Agent)
                combination.objs[nbrIndx];

            double nbrProperty = neighbour.
                diversity[property];
            if(nbrProperty>combProperty){
```

```
                        combProperty += nbrProperty;
                    }
                }
                combProperty *= depreciation;

                /* the agent will also consume a share
                   of the product of
                 * each neighbour;
                 * here we add each of the neighbours'
                    estimated
                 * property for the next period */
                for(int nbrCombIndex=0;
                        nbrCombIndex<maxRelationships;
                        nbrCombIndex++) {
                    Agent neighbour = (Agent)
                       combination.objs[nbrCombIndex];
                    if(neighbour!=newNeighbour) {
                        combProperty +=
                                neighbour.
                                   estimatedDiversity[
                                   property];
                    }
                    else {
                        combProperty += newNbrDiversity
                           [property];
                    }
                }
                combinationValue += combProperty;
            }
            combinationValue *= invNumAgents;

            /* If this combination generates a higher
               utility than the
             * existing best, then this will replace it
                */
            if(combinationValue>bestCombinationValue) {
                bestCombination[nodeIndex] =
                   combination;
                bestCombinationValue = combinationValue
                   ;
                agent.estimatedUtility =
                   combinationValue;
            }
        }
    }
```

```
            /* If we've got this far, then the edge has been
               approved by both
             * neighbours, which means it needs to be added,
                and the deprecated
             * links removed */
            agents[0].neighbours.add(agents[1]);
            agents[1].neighbours.add(agents[0]);
            for(int nodeIndex=0;nodeIndex<2;nodeIndex++){
                for(int neighbIndex=0;
                        neighbIndex<agents[nodeIndex].
                           neighbours.numObjs;
                        neighbIndex++){
                    if(!bestCombination[nodeIndex]
                            .contains(agents[nodeIndex].
                               neighbours.objs[neighbIndex])){
                        if(neighboursToRemove[agents[nodeIndex
                            ].index]==null){
                            neighboursToRemove[agents[nodeIndex
                                ].index]
                                    = new Bag();
                        }
                        neighboursToRemove[agents[nodeIndex].
                            index]
                                .add(agents[nodeIndex].
                                   neighbours.objs[neighbIndex
                                   ]);
                        if(neighboursToRemove[((Agent)agents[
                            nodeIndex]
                                .neighbours.objs[neighbIndex]).
                                   index]==null){
                            neighboursToRemove[((Agent)agents[
                                nodeIndex]
                                    .neighbours.objs[
                                       neighbIndex]).index]
                                    = new Bag();
                        }
                        neighboursToRemove[((Agent)agents[
                            nodeIndex].neighbours.objs[
                            neighbIndex]).index]
                                .add(agents[nodeIndex]);
                        edgesRemoved++;
                        break;
                    }
                }
            }
        }
```

```
        for(int i=0; i<numAgents; i++) {
            Agent subject = (Agent)allNodes.objs[i];
            subject.removeNeighbours(neighboursToRemove[i]);
            /* Now that topological changes are complete, we'll
                record this
             * neighbours Bag as the oldNeighbours Bag for the
                next period */
            subject.oldNeighbours = new Bag(subject.neighbours)
               ;
        }

        newEdges.clear();
    }

    /* a recursive method to construct combinations */
    void getCombinations(Bag superSet, int lastPick, int
       remainingSpaces,
            Bag combination, Bag combinations) {
        if(remainingSpaces == 0) {
            combinations.add(combination);
            return;
        }
        int thisChoiceBound = superSet.numObjs -
           remainingSpaces + 1;
        int remainingSpacesNow = remainingSpaces - 1;
        for(int pickIndex = lastPick+1; pickIndex<
           thisChoiceBound; pickIndex++) {
            Object obj = superSet.objs[pickIndex];
            Bag combCtd = new Bag(combination);
            combCtd.add(obj);
            getCombinations(superSet, pickIndex,
               remainingSpacesNow, combCtd,
                        combinations);
        }
    }

    void discoverComponents() {
        // refresh the bag of components
        components = new Bag();

        // node by node, check discovered components
           sequentially for membership.
        // if absent, it becomes the start node of a new
           component exploration
        for(int i=0; i<numAgents; i++) {
            Agent agent = (Agent)allNodes.objs[i];
```

```
            boolean componentFound = false;

            for(int componentIndex=0;
                    componentIndex<components.numObjs;
                    componentIndex++) {
                Bag existingComponent = (Bag)components.objs[
                    componentIndex];

                if(existingComponent.contains(agent)) {
                    componentFound = true;
                }
            }

            if(componentFound) continue;

            // if an existing component isn't that of the agent
                , we start a new
            // one and explore
            Bag newComponent = new Bag();
            newComponent.add(agent);
            components.add(newComponent);
            agent.component = newComponent;
            exploreComponent(agent,newComponent);
        }
    }

    void exploreComponent(Agent subject, Bag component) {
        Bag neighbours = subject.neighbours;

        for(int i=0; i<neighbours.numObjs; i++) {
            Agent neighbour
                    = (Agent)neighbours.objs[i];
            if(!component.contains(neighbour)) {
                component.add(neighbour);
                neighbour.component = component;
                exploreComponent(neighbour,component);
            }
        }
    }

    int shortestPath(int pathLength, Object destinationNode,
        Object avoidNode,
            Bag visitedNodes, Bag oldTier) {
        pathLength++;
```

```
Bag newTier = new Bag();
for(int i=0; i<oldTier.numObjs; i++) {
    Agent agent = (Agent)oldTier.objs[i];
    Bag neighbours = agent.neighbours;

    // first we'll go through the neighbours From which
        the In edges come
    for(int j=0; j<neighbours.numObjs; j++) {
        Agent neighbour = (Agent)neighbours.objs[j];

        // we want to make sure that neighbour is not
           the original node,
        // avoidNode
        // we only want to continue if we have not
           already found a
        // faster route to that node: if it is not in a
            lower tier
        if(!(neighbour.equals(avoidNode))
                && (!visitedNodes.contains(neighbour)))
                    {
            // we check whether we've reached the
               destination
            if(neighbour.equals(destinationNode) ==
               true) {
                return pathLength;
            }
            // if not, we extend the path so far by the
                current node,
            // and search on from there
            else {
                newTier.add(neighbour);
                visitedNodes.add(neighbour);
            }
        }
    }
}

// having built the next tier we want to pass it to a
   deeper recurrence
// of the same algorithm, all the while keeping track
   of the number of
// tiers
if(!newTier.isEmpty()) {
    pathLength = shortestPath(pathLength,
       destinationNode,avoidNode,
            visitedNodes,newTier);
```

```
            return pathLength;
        }
        else {
            // this shouldn't happen
            System.out.println("No path found to " + ((Agent)
                destinationNode).index
                    + " from " + ((Agent)avoidNode).index);
            return numAgents;
        }
    }

    /* public void updateStats()
     * checks various network statistics
     * we want this to be run only on request, to update and
        report the stats
     * for the network
     * so we want clustering distribution, degree distribution,
         centrality,
     * network clustering and support,
     * and component sizes -- possibly the component size
        distribution
     */
    public void updateStats() {
        averageDegree = 0;
        averageClustering = 0;
        averageEdgeAge = 0;
        clustering = 0;
        support = 0;
        degreeDistribution = new int[numAgents];
        clusteringDistribution = new double[numAgents];

        numRelationships = 0;
        averageProduct = .0;
        averageDiversityMeasure = 0.0;
        int maxTotalRelationships = 0;

        int largestComponentSize = 0;
        int secondComponentSize = 0;
        int totalPotentialTriads = 0;
        int completeTriads = 0;
        int supportedLinks = 0;
        int potentialSupport = 0;

        discoverComponents();

        for(int i=0; i<numAgents; i++) {
```

```
Agent agent = (Agent)allNodes.objs[i];
Bag agentRelationships = agent.neighbours;

numRelationships += agentRelationships.size();
maxTotalRelationships += maxRelationships;

averageProduct += agent.product;

Bag component = (Bag)agent.component;
int componentSize = component.numObjs;
if(componentSize>largestComponentSize) {
    secondComponentSize = largestComponentSize;
    secondComponent = largestComponent;

    largestComponentSize = componentSize;
    largestComponent = component;
}
else if(componentSize>secondComponentSize
        && componentSize!=largestComponentSize) {
    secondComponentSize = componentSize;
    secondComponent = component;
}

int degree = agent.neighbours.numObjs;
degreeDistribution[i] = degree;
averageDegree = averageDegree + (double)degree;

int commonNeighbours = 0;
int potentialTriads = 0;
boolean[] supportFound = new boolean[
   agentRelationships.numObjs];

for(int j=0; j<agentRelationships.numObjs; j++) {
    Agent neighbour = (Agent)agentRelationships.
       objs[j];
    Bag neighbourRelationships = neighbour.
       neighbours;

    // support is a link property; one and only one
        supporting node
    // (edge-pair) need be found to qualify that
       edge as supported
    // the way it's set up here you will count each
        link twice -
    // once with each participant as the subject,
       agent
```

```
        supportFound[j] = false;
        potentialSupport++;

        kLoop:for(int k=0; k<j; k++) {
            Agent otherNeighbour = (Agent)
                agentRelationships.objs[k];
            Bag otherNeighbourNeighbours =
                otherNeighbour.neighbours;

            // this pair of neighbours represents a
                potential triad,
            // regardless of whether they actually are
            potentialTriads++;

            for(int l=0; l<otherNeighbourNeighbours.
                numObjs; l++) {
                Agent possCommonNeighbour
                        = (Agent)
                            otherNeighbourNeighbours.
                            objs[l];
                if(neighbourRelationships.contains(
                    possCommonNeighbour)) {
                    if(supportFound[j]==false) {
                        supportedLinks++;
                        supportFound[j] = true;
                    }
                    if(supportFound[k]==false) {
                        supportedLinks++;
                        supportFound[k] = true;
                    }
                    // as this pair of neighbours are
                        themselves
                    // neighbours, the clustering will
                        increase by 1
                    commonNeighbours++;
                    break;
                }
            }
        }
    }
    totalPotentialTriads = totalPotentialTriads +
        potentialTriads;
    if(potentialTriads>0) clusteringDistribution[i] = (
        double)commonNeighbours/(double)potentialTriads;
    averageClustering = averageClustering +
        clusteringDistribution[i];
```

```
        completeTriads = completeTriads + commonNeighbours;
    }

    averageEdgeAge = averageEdgeAge / numRelationships;
    numRelationships = numRelationships/2;

    completeness =
            (double) numRelationships /
               maxTotalRelationships;
    averageProduct = averageProduct / numAgents;
    averageDiversityMeasure = averageDiversityMeasure /
       numAgents;

    relationshipsBroken = edgesRemoved;
    edgesRemoved = 0;

    // calculate clustering and support
    if(totalPotentialTriads>0) clustering
            = (double) completeTriads / (double)
               totalPotentialTriads;
    if(potentialSupport>0) support
            = (double)supportedLinks / (double)
               potentialSupport;

    // turn sums into averages, and absolute sizes into
       population shares
    averageDegree = averageDegree/numAgents;
    averageClustering = averageClustering/numAgents;
    largestComponentShare = (double)largestComponentSize/
       numAgents;
    secondComponentShare = (double)secondComponentSize/
       numAgents;

    // if the network is connected, then we want to check
       the average
    // shortest path length
    averageShortestPath = 0.0;
    for(int i=0; i<largestComponentSize; i++) {
        Agent startAgent = (Agent)largestComponent.objs[i];
        for(int j=0; j<i; j++) {
            Agent finishAgent = (Agent)largestComponent.
               objs[j];

            Bag oldTier = new Bag();
            oldTier.add(startAgent);
```

```
                averageShortestPath += shortestPath(0,
                    finishAgent,
                        startAgent,new Bag(oldTier),oldTier);
            }
        }
        averageShortestPath = averageShortestPath * 2.0 / (
            numAgents*(numAgents-1));
    }
}
```

TradeSafer.java

```
/*
 Copyright 2011 by Tom Wilkinson and Cardiff University
 Licensed under the Academic Free License version 3.0
 See the file "LICENSE" for more information
*/
package anarchytrade;
import java.io.*;
import java.util.Date;
import java.text.DateFormat;
import java.text.SimpleDateFormat;

/**Houses a main method that runs a series of simulations of
    an exchange network
* model, with endogenous topology, over a range of parameters
    for the
* probability of searching globally for a new relationship
*/
public class TradeSafer {
    static String outputFileName = "compstd.txt";
    static String outputPathName = "C:\\simulate\\safer";
    static File outputFile;
    static File outputPath;
    static PrintWriter out;

    // Simulation parameters
    static int defaultNumSims = 10;
    static int defaultSimDuration = 50000;
    static int defaultNumAgents = 100;
    static int defaultOutputPeriod = 100;

    // agents' parameters
    static int defaultMaxRelationships = 4;
    static double defaultDepreciation = 0.9;
    static double defaultInnovation = 1.0;
```

```
// socialNetwork's parameters
static boolean defaultLocalSearch = true;
static double defaultGlobalSearchProb = 0.5;
static double defaultInterval = 0.25;
static int defaultIncrements = 5;
static int defaultSearchPeriod = 10;

public static void main(String[] args)
{
    // print help?
    if (keyExists("-help", args, 0))
        {
        System.err.println(
            "Format:␣␣␣␣␣␣␣␣␣␣␣java␣-jar␣mason.jar␣[-
                numAgents␣n]␣[-riskAversion␣r]␣[-favourShare
                ␣f]␣\\\n" +
            "␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣[-help]␣[-repeat␣R]␣[-
                seed␣S]␣[-until␣U]␣\\\n" +
            "␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣[-for␣F]␣[-time␣T]␣[-
                docheckpoint␣D]␣[-checkpoint␣C]␣\n\n" +
            "-help␣␣␣␣␣␣␣␣␣␣␣␣Shows␣this␣message␣and␣exits
                .\n\n" +
            "-numAgents␣␣␣␣␣␣␣␣int␣>␣1:␣the␣number␣of␣
                agents␣in␣the␣simulation␣\n\n" +
            "-maxRelationships␣int␣>␣1:␣the␣maximum␣number␣
                of␣neighbours␣allowed␣\n\n" +
            "-localSearch␣␣␣␣␣␣boolean␣in␣{0,1}:␣whether␣
                links␣to␣neighbours␣of␣neighbours␣are\n" +
            "␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣offered␣with␣higher␣
                probability␣than␣those␣to␣other␣agents\n\n"
                +
            "-globalSearchProb␣0<␣double␣<1:␣the␣initial␣
                probability␣with␣which␣a␣new␣neighbour␣is␣
                met␣each␣period\n\n" +
            "-interval␣␣␣␣␣␣␣␣␣the␣interval␣above␣this␣over
                ␣which␣probabilities␣will␣be␣tried␣\n\n" +
            "-innovation␣␣␣␣␣␣␣double␣>0:␣the␣ammount␣of␣
                the␣good␣produced␣by␣each␣agent␣each␣period␣
                \n\n" +
            "-depreciation␣␣␣␣␣0<␣double␣<1:␣the␣rate␣at␣
                which␣products␣lose␣value␣with␣distance␣
                traveled␣\n\n" +
            "-searchPeriod␣␣␣␣␣int␣>0:␣the␣number␣of␣
                periods␣between␣each␣exchange␣event␣\n\n" +
            "-outputPeriod␣␣␣␣␣␣␣the␣number␣of␣steps␣
                between␣writing␣statistics␣to␣the␣output␣
```

```
                file\n\n" +
            "-numSims          Long value > 0: Runs the job
                R times.   Unless overridden by a\n" +
            "                  checkpoint recovery (see -
                checkpoint), the random seed for\n" +
            "                  each job is the provided -
                seed plus the job# (starting at 0).\n" +
            "                  Default: runs once only: job
                number is 0.\n\n" +
            "-seed             Long value not 0: the random
                number generator seed, unless \n" +
            "                  overridden by a checkpoint
                recovery (see -checkpoint).\n" +
            "                  Default: the system time in
                milliseconds.\n\n" +
            "-simDuration      Long value >= 0: the
                simulation must stop when N\n" +
            "                  simulation steps have
                transpired.\n" +
            "                  Default: don't stop.\n\n");
        System.exit(0);
        }

    java.text.NumberFormat n = java.text.NumberFormat.
        getInstance();
    n.setMinimumFractionDigits(0);
    System.err.println("Starting simulation");

    // set numAgents
    int numAgents = defaultNumAgents;
    String numAgents_s = argumentForKey("-numAgents", args,
        0);
    if (numAgents_s != null)
        try {
            numAgents = Integer.parseInt(numAgents_s);
            if (numAgents < 2) throw new Exception();
            }
        catch (Exception e) {
            throw new RuntimeException("Invalid 'numAgents'
                value: "
                    + numAgents_s + ", must be greater than
                        1");
            }
    // set maxRelationships
    int maxRelationships = defaultMaxRelationships;
```

```
String maxRelationships_s = argumentForKey("-
    maxRelationships", args, 0);
if (maxRelationships_s != null)
    try {
        maxRelationships = Integer.parseInt(
            maxRelationships_s);
        if (maxRelationships < 2) throw new Exception()
            ;
        }
    catch (Exception e) {
        throw new RuntimeException("Invalid␣'
            maxRelationships'␣value:␣"
                + maxRelationships_s + ",␣must␣be␣
                    greater␣than␣1");
        }
// set localSearch
boolean localSearch = defaultLocalSearch;
String localSearch_s = argumentForKey("-localSearch",
    args, 0);
if (localSearch_s != null)
    try {
        localSearch = Boolean.parseBoolean(
            localSearch_s);
        }
    catch (Exception e) {
        throw new RuntimeException("Invalid␣'
            localSearch'␣value:␣"
                + localSearch_s + ",␣must␣be␣0␣or␣1");
        }
// set globalSearchProb
double globalSearchProb = defaultGlobalSearchProb;
String globalSearchProb_s = argumentForKey("-
    globalSearchProb", args, 0);
if (globalSearchProb_s != null)
    try {
        globalSearchProb = Double.parseDouble(
            globalSearchProb_s);
        if (globalSearchProb < 0.0 || globalSearchProb
            > 1.0)
            throw new Exception();
        }
    catch (Exception e) {
        throw new RuntimeException("Invalid␣'
            globalSearchProb'␣value:␣"
                + globalSearchProb_s + ",␣must␣lie␣in␣
                    the␣unit␣interval");
```

```
        }
// set exploration interval for globalSearchProb
double interval = defaultInterval;
String interval_s = argumentForKey("-interval", args,
   0);
if (interval_s != null)
    try {
        interval = Double.parseDouble(interval_s);
        if (interval < 0.1 || interval > 1.0)
            throw new Exception();
        }
    catch (Exception e) {
        throw new RuntimeException("Invalid␣'interval'␣
           value:␣"
                + interval_s + ",␣must␣lie␣in␣the␣unit␣
                   interval");
        }
// set number of increments for globalSearchProb
int increments = defaultIncrements;
String numIncrements_s = argumentForKey("-increments",
   args, 0);
if (numIncrements_s != null)
    try {
        increments = Integer.parseInt(numIncrements_s);
        if (increments < 0)
            throw new Exception();
        }
    catch (Exception e) {
        throw new RuntimeException("Invalid␣'increments
           '␣value:␣"
                + numIncrements_s + ",␣must␣be␣positive
                   ");
        }
// set searchPeriod
int searchPeriod = defaultSearchPeriod;
String meetingRate_s = argumentForKey("-searchPeriod",
   args, 0);
if (meetingRate_s != null)
    try {
        searchPeriod = Integer.parseInt(meetingRate_s);
        if (searchPeriod < 0)
            throw new Exception();
        }
    catch (Exception e) {
        throw new RuntimeException("Invalid␣'
           searchPeriod'␣value:␣"
```

```
                + globalSearchProb_s + ", must be
                    positive");
        }
// set depreciation
double depreciation = defaultDepreciation;
String depreciation_s = argumentForKey("-depreciation",
    args, 0);
if (depreciation_s != null)
    try {
        depreciation = Double.parseDouble(
            depreciation_s);
        if (depreciation < 0.0 || depreciation > 1.0)
            throw new Exception();
        }
    catch (Exception e) {
        throw new RuntimeException("Invalid '
            depreciation' value: "
                + depreciation_s + ", must lie in the
                    unit interval");
        }
// set innovation
double innovation = defaultInnovation;
String innovation_s = argumentForKey("-innovation",
   args, 0);
if (innovation_s != null)
    try {
        innovation = Double.parseDouble(innovation_s);
        if (innovation < 0.0) throw new Exception();
        }
    catch (Exception e) {
        throw new RuntimeException("Invalid 'innovation
            ' value: "
                + innovation_s + ", must be positive");
        }
long seed = System.currentTimeMillis();
String seed_s = argumentForKey("-seed", args, 0);
if (seed_s != null)
    try
        {
        seed = Long.parseLong(seed_s);
        if (seed == 0) throw new Exception();
        }
    catch (Exception e)
        {
        throw new RuntimeException("Invalid 'seed'
            value: "
```

```
                + seed_s + ", must be a non-zero
                    integer, or nonexistent "
                + "to seed by clock time");
        }
int simDuration = defaultSimDuration;
String simDuration_s = argumentForKey("-simDuration",
    args, 0);
if (simDuration_s != null)
    try {
        simDuration = Integer.parseInt(simDuration_s);
        if (simDuration < 0) throw new Exception();
        }
    catch (Exception e) {
        throw new RuntimeException("Invalid '
            simDuration' value: "
                + simDuration_s + ", must be an integer
                    >= 0");
        }
int numSims = defaultNumSims;
String numSims_s = argumentForKey("-numSims", args, 0);
if (numSims_s != null)
    try {
        numSims = Integer.parseInt(numSims_s);
        if (numSims <= 0) throw new Exception();
        }
    catch (Exception e) {
        throw new RuntimeException("Invalid 'numSims'
            value: "
                + numSims + ", must be a positive
                    integer");
        }
int outputPeriod = defaultOutputPeriod;
String outputPeriod_s = argumentForKey("-outputPeriod",
    args, 0);
if (outputPeriod_s != null)
    try {
        outputPeriod = Integer.parseInt(outputPeriod_s)
            ;
        if (outputPeriod <= 0) throw new Exception();
        }
    catch (Exception e) {
        throw new RuntimeException("Invalid '
            outputPeriod' value: "
                + outputPeriod + ", must be a positive
                    integer");
        }
```

```
/* Set up the output universals */
java.text.NumberFormat rateFormat = java.text.
    NumberFormat.getInstance();
rateFormat.setMaximumFractionDigits(5);
rateFormat.setMinimumIntegerDigits(1);
DateFormat dateFormat = new SimpleDateFormat("MMdd");
Date date = new Date();

outputPath = new File(outputPathName
        + "." + numAgents + "." + maxRelationships + "
            \\");
try {
    if(!outputPath.exists()) outputPath.mkdirs();
}
catch (Exception e) {
    System.err.printf("Could␣not␣create␣output␣
        directory");
}

// Now to get underway, exploring the range of global
    search probabilities
double incrementSize = interval / (double)increments;
for(int increment=0; increment<increments; increment++)
    {
    /* set the global search probability for this run
        */
    double currentProb = globalSearchProb
            + (double)increment*incrementSize;
    /* get many simulations' worth of data */
    for(int sim=0; sim<numSims; sim++){
        // instantiate the social network
        SocialNetwork society = new SocialNetwork(
            localSearch, numAgents,
                maxRelationships, searchPeriod,
                    currentProb,
                depreciation, innovation);

        // instantiate the agents
        for(int index=0; index<numAgents; index++) {
            Agent agent = new Agent(numAgents, index,
                maxRelationships,
                    depreciation, innovation);
            society.allNodes.add(agent);
        }
```

```
/* Set up the output specifics */
String fileName = outputPathName +"."+
   numAgents
        +"."+ maxRelationships +"\\"+
            globalSearchProb +"_"
        + depreciation +"_"+ sim +"_"+
            dateFormat.format(date)
        +"_"+ outputFileName;
try (BufferedWriter out
        = new BufferedWriter(new FileWriter(
            fileName))) {

    // Let's print out the whole parameter list
    out.write("Output␣parameters:␣outputPeriod␣
       "+outputPeriod+"\n");
    out.write("Social␣network␣parameters:␣
       numAgents=" + numAgents +
            ",␣searchPeriod=" + searchPeriod +
            ",␣depreciation=" + depreciation +
            ",␣globalSearchProb=" +
               globalSearchProb+"\n");
    out.write("Agent␣parameters:␣
       maxRelationships="
            +maxRelationships+",␣innovation="+
               innovation+"\n");
}
catch (FileNotFoundException e) {
    System.err.println("FileNotFoundException:␣
       " + e.getMessage());
}
catch (IOException e) {
    System.err.println("Caught␣IOException:␣" +
        e.getMessage());
}

/* now a simulation is run */
int outputWait = outputPeriod;
int searchWait = searchPeriod;
for(int steps=0; steps<simDuration; steps++){
    /* new neighbours are offered and judged */
    if(--searchWait < 0){
        society.neighbourSearch();
        society.neighbourChoose();
        searchWait = searchPeriod;
    }
```

```
                /* agents exchange, and learn about one
                   another */
                society.exchangeTransfers();

                /* output the statistics describing current
                    network topology */
                if(--outputWait < 0){
                    society.updateStats();
                    out.print("Average␣degree,␣" + society.
                       averageDegree + ",␣");
                    out.print("Average␣clustering,␣"+
                       society.averageClustering + ",␣");
                    out.print("Clustering␣," + society.
                       clustering + ",␣");
                    out.print("Support,␣" + society.support
                        + ",␣");
                    out.print("Largest␣Component,␣"+society
                       .largestComponentShare + ",␣");
                    out.print("Second␣Component,␣"+society.
                       secondComponentShare + ",␣");
                    out.print("Avg␣shortest␣path,␣"+society
                       .averageShortestPath + ",␣");
                    out.print("Relationships,␣"+society.
                       numRelationships + ",␣");
                    out.print("Completeness,␣" + society.
                       completeness + ",␣");
                    out.print("Broken␣links,␣"+society.
                       relationshipsBroken);
                    out.println();
                    outputWait = outputPeriod;
                }
            }
            if (out != null) {
                System.out.println("Closing␣PrintWriter");
                out.print("Simulation␣Complete");
                out.close();
            }
            else {
                System.out.println("PrintWriter␣not␣open");
            }
            System.out.println("Quit");
            seed++;
            }
        }
    System.exit(0);
    }
```

```
    static String argumentForKey(String key, String[] args, int
        startingAt)
        {
        for(int x=0;x<args.length-1;x++)  // key can't be the
            last string
             if (args[x].equalsIgnoreCase(key))
                 return args[x + 1];
        return null;
        }

    static boolean keyExists(String key, String[] args, int
       startingAt)
        {
        for(int x=0;x<args.length;x++)  // key can't be the
            last string
             if (args[x].equalsIgnoreCase(key))
                 return true;
        return false;
        }
}
```

5.A.2 Model version with full information

This model is identical except for one method in the social network object.

NetworkFullInf.java

```
/*
 Copyright 2011 by Tom Wilkinson and Cardiff University
 Licensed under the Academic Free License version 3.0
 See the file "LICENSE" for more information
*/
package anarchyfullinf;
import util.*;

/* Anarchy Trade model allows agents to predict the effects of
    adding neighbours
* on those neighbours' diversities!
*/
public class NetworkFullInf {
    Bag allNodes;
    MersenneTwisterFast random;

    final boolean localSearch;
```

```
final int numAgents;
final int maxRelationships;
final int exchangeRate;
final double globalSearchProb;

final double depreciation;
final double innovation;

final double invNumAgents;

Bag newEdges;

int edgesRemoved;
boolean statsTaken;
Bag components;
Bag largestComponent;
Bag secondComponent;

// instantaneous network statistics:
double averageDegree;
int[] degreeDistribution;
double averageClustering;
double[] clusteringDistribution;
double clustering;
double support;
double[] centrality;
double largestComponentShare;
double secondComponentShare;
double averageShortestPath;
double averageEdgeAge;

Bag parents;

// dynamic network statistics:
double averageProduct;
double averageDiversityMeasure;
int numRelationships;
double completeness;
int relationshipsBroken;

public NetworkFullInf(boolean localSearch, int numAgents,
   int maxRelationships,
        int exchangeRate, double globalSearchProb, double
           depreciation,
        double innovation) {
```

```
        this.localSearch = localSearch;

        this.numAgents = numAgents;
        this.maxRelationships = maxRelationships;
        this.exchangeRate = exchangeRate;
        this.globalSearchProb = globalSearchProb;
        this.depreciation = depreciation;
        this.innovation = innovation;

        invNumAgents = 1.0/(double)numAgents;

        newEdges = new Bag();

        random = new MersenneTwisterFast();
    }

    // Transfers are now called in from the various agents, and
        then redistributed
    public void exchangeTransfers() {
        double[][] diversities = new double[numAgents][
            numAgents];

        // first, let's call in each of the updated diversity
            arrays
        for(int agentIndex=0; agentIndex<numAgents; agentIndex
            ++) {
            Agent agent = (Agent)allNodes.objs[agentIndex];
            diversities[agentIndex] = agent.receiveTransfers();
        }

        // now, with all agents' diversities updated, the
            original diversities
        // are no longer needed and can be replaced
        for(int agentIndex=0; agentIndex<numAgents; agentIndex
            ++) {
            Agent agent = (Agent)allNodes.objs[agentIndex];
            agent.diversity = diversities[agentIndex];
        }
    }


    /* After transfers but before choices, this provides new
       neighbours to those
     * agents that can sustain them
     */
    public void neighbourSearch() {
```

```
Bag classifieds = new Bag(allNodes);

/* local search should happen first, to reduce the
   possibility
 * that there are no viable local search choices left
    in classifieds:
 * - if there is a clique there could still be no local
    search choices */

Agent agent =
        (Agent)classifieds.objs[random.nextInt(
           classifieds.numObjs)];

// we create a new bag to keep track of neighbours we'
   ve visited
// in local search
Bag neighbours = new Bag(agent.neighbours);

// now a specific classifieds bag stopping us selecting
// inappropriate new links for this particular agent
Bag tempClassifieds = new Bag(classifieds);
tempClassifieds.remove(agent);
tempClassifieds.removeAll(neighbours);
if(tempClassifieds.isEmpty()) {
    classifieds.remove(agent);
}

// with a given probability a new neighbour is found by
    global search
if(random.nextBoolean(globalSearchProb)) {
    int potentialClassified =
            random.nextInt(tempClassifieds.numObjs);
    Agent potentialNeighbour = (Agent)tempClassifieds
            .objs[potentialClassified];
    /* now we add these agents as a pair to the bag of
       new edges
     * to check out */
    newEdges.add(new Agent[]{agent,potentialNeighbour})
       ;

    classifieds.remove(agent);
    classifieds.remove(potentialNeighbour);
}
// otherwise local search is used - may not be
   successful!
else if(localSearch) {
```

```
            // first we randomly choose a neighbour from whom
               to pick a
            // common neighbour
            while(!neighbours.isEmpty()) {
                int neighbourIndex = random.nextInt(neighbours.
                   numObjs);
                Agent viaNeighbour = (Agent)neighbours.objs[
                   neighbourIndex];

                Bag potentials = new Bag(viaNeighbour.
                   neighbours);
                potentials.remove(agent);

                // now we randomly search through that agent's
                // neighbours
                while(!potentials.isEmpty()) {
                    int potentialIndex = random.nextInt(
                       potentials.numObjs);
                    Agent potential = (Agent)potentials.objs[
                       potentialIndex];

                    if(tempClassifieds.contains(potential)) {
                        newEdges.add(new Agent[]{agent,
                            potential});

                        classifieds.remove(agent);
                        classifieds.remove(potential);
                        return;
                    }
                    else potentials.remove(potential);
                }
                neighbours.remove(viaNeighbour);
            }
        }
    }

    /* In this hyperopic version of the model, the agent party
       to the offered
     * link will actually see the shortest paths to every other
         agent, and will
     * calculate their expected utility according to these */
    public void neighbourChoose()
    {
        Bag[] neighboursToRemove = new Bag[numAgents];
        int numNewEdges = newEdges.numObjs;
```

```
newEdgeLoop: for(int newEdgeIndex=0; newEdgeIndex<
    numNewEdges; newEdgeIndex++)
{
    Agent[] agents = (Agent[])newEdges.objs[
        newEdgeIndex];
    Bag weakLinks = new Bag();
    Agent[] rejects = new Agent[2];

    /* Like the myopic model, we'll assume that the
        agents will ditch
     * the independently least valuable of their
        neighbours in favour
     * of the other party to the link, IF that link
        adds value */
    for(int nodeIndex=0; nodeIndex<2; nodeIndex++)
    {
        Agent agent = agents[nodeIndex];

        Bag neighbours = new Bag(agent.neighbours);

        int numNeighbours = neighbours.numObjs;
        int maxNeighbours = maxRelationships-1;
        /* if the agent can sustain all links, then
            they will - it is
         * assumed that all links will provide positive
            value or defection
         * will just lead to equal value */
        if(numNeighbours<maxRelationships)
        {
            continue;
        }
        /* if there are too many neighbours to sustain,
            then we'll assess
         * the value of all combinations (size of
            maxRelationships-1) */
        Bag combination = new Bag();
        Bag combinations = new Bag();
        getCombinations(neighbours, -1, maxNeighbours,
            combination,
                combinations);
        int numCombinations = combinations.size();
        Bag bestCombination = new Bag();
        double bestCombinationValue = 0.0;

        /* now to explore all the possible combinations
            of neighbours */
```

```
for(int combinationIndex=0;
        combinationIndex<numCombinations;
        combinationIndex++)
{
    combination = (Bag)combinations.objs[
       combinationIndex];

    /* now an implementation with non-separable
        utility:
     * - as the effects of the edge's other
        party are not being
     *   considered, there is no need to
        consider different
     *   different initial and subsequent
        neighbourhoods */
    double combinationValue = 0.0;

    /* Now to map out the shortest paths in the
        network, ignoring
     * the neighbour that isn't included in
        this combination:
     * - this will require the shortest path*/
    for(int property=0; property<numAgents;
       property++)
    {
        Agent propertyProducer = (Agent)
           allNodes.objs[property];
        Bag visited = new Bag(combination);
        visited.add(agent);
        int shortestPath;
        if(property==agent.index) shortestPath
           = 0;
        else if(combination.contains(
           propertyProducer))
            shortestPath = 1;
        else shortestPath = shortestPath(1,
           propertyProducer,
                agent,visited, new Bag(
                   combination));
        if(shortestPath>0)
        {
            double pathValue = innovation;
            double stepWeight = depreciation;
            for(int step=0; step<shortestPath;
               step++)
                pathValue *= stepWeight;
```

```
                        combinationValue += pathValue;
//                          combinationValue += 1 - Math.exp
    (xenophilia*pathValue);
                    }
                }
                combinationValue *= invNumAgents;

                /* If this combination generates a higher
                   utility than the
                 * existing best, then this will replace it
                    */
                if(combinationValue>bestCombinationValue)
                {
                    bestCombination = combination;
                    bestCombinationValue = combinationValue
                        ;
                }
            }
            /* Having found the best combination of
               neighbours for this agent,
             * we find the neighbour that it does not
               contain */
            for(int nbrIndex=0; nbrIndex<neighbours.numObjs
               ; nbrIndex++)
            {
                if(!bestCombination.contains(neighbours.
                   objs[nbrIndex]))
                {
                    rejects[nodeIndex] = (Agent)neighbours.
                       objs[nbrIndex];
                    agent.guessRejectedInd = rejects[
                       nodeIndex].index;
                }
            }
        }

        /* the following loop will assess the options for
           each agent,
         * continuing the thisEdge loop if either
           dissapproves, but allowing
         * the link to be added if both approve */
        Bag[] bestCombination = new Bag[2];
        for(int nodeIndex=0; nodeIndex<2; nodeIndex++)
        {
            Agent agent = agents[nodeIndex];
            Agent newNeighbour = agents[(nodeIndex+1)%2];
```

```
Agent nbrReject = rejects[(nodeIndex+1)%2];

Bag neighbours = new Bag(agent.neighbours);
neighbours.add(newNeighbour);

int numNeighbours = neighbours.numObjs;
/* if the agent can sustain all links, then
   they will - it is
 * assumed that all links will provide positive
     value or defection
 * will just lead to equal value */
if(numNeighbours<=maxRelationships) continue;

/* if there are too many neighbours to sustain,
    then we'll assess
 * the value of all combinations (size of
    maxRelationships) */
Bag combination = new Bag();
Bag combinations = new Bag();
getCombinations(neighbours, -1,
   maxRelationships, combination,
        combinations);
int numCombinations = combinations.size();
double bestCombinationValue = 0.0;

/* now to explore all the possible combinations
    of neighbours */
for(int combinationIndex=0;
        combinationIndex<numCombinations;
        combinationIndex++)
{
    combination = (Bag)combinations.objs[
       combinationIndex];
    /* For each combination, we need to find
       the value to the agent
     * of this set of edges:
     * - The neighbourhoods of both the agent
       and the new
     *   neighbour will change, but only after
       the first exchange
     *   HOWEVER WE WILL ASSUME THE FORMER
       NEIGHBOUR's PROPERTIES
     *   THAT WOULD HAVE BEEN CARRIED OVER FOR
       THE FIRST EXCHANGE
     *   WILL BE OFFSET BY A COST TO NEW
       RELATIONSHIPS */
```

```
double combinationValue = 0.0;

/* Now to add the transfers for each of the
    neighbours in
 * this combination, assuming non-separable
    utility
 * - which means summing properties */
for(int property=0; property<numAgents;
   property++)
{
    Agent propertyProducer = (Agent)
       allNodes.objs[property];
    int shortestPath = -1;
    if(property==agent.index) shortestPath
       = 0;
    else if(combination.contains(
       propertyProducer))
        shortestPath = 1;
    else if(combination.contains(
       newNeighbour))
    {
        /* If the property isn't that of
           the agent or a
         * neighbour (in this combination),
             then, when the
         * new neighbour is in the
            combination, could either
         * be
         * - a path not via the
            newNeighbour
         * - or, a path via the
            newNeighbour (restricted by
         *   the loss of their rejected
            node) */
        Bag firstTier = new Bag(combination
           );
        firstTier.remove(newNeighbour);
        Bag visited = new Bag(combination);
        visited.add(agent);
        shortestPath = shortestPath(1,
           propertyProducer,
                agent,visited, firstTier);
        /* now let's see if there's a
           faster path via the
         * new neighbour */
```

```
                    firstTier = new Bag(newNeighbour.
                        neighbours);
                    firstTier.remove(nbrReject);
                    visited = new Bag(firstTier);
                    visited.add(newNeighbour);
                    visited.add(agent);
                    int shortestNewPath =
                            shortestPath(2,
                                propertyProducer,agent,
                            visited, firstTier);
                    if(shortestNewPath > 0
                            && (shortestNewPath <
                                shortestPath
                            || shortestPath < 0)){
                        shortestPath = shortestNewPath;
                    }
                }
                else {
                    /* If this combination is the
                        original neighbourhood
                     * of the agent, then there is no
                         need to worry
                     * about the newNeighbour */
                    Bag visited = new Bag(combination);
                        visited.add(agent);
                        shortestPath = shortestPath(1,
                            propertyProducer,
                                agent,visited, new Bag(
                                    combination));
                }
                if(shortestPath>=0){
                    double pathValue = innovation;
                    double stepWeight = depreciation;
                    for(int step=0; step<shortestPath;
                        step++)
                        pathValue *= stepWeight;
                    combinationValue += pathValue;
                }
            }
            combinationValue *= invNumAgents;

            /* If this combination generates a higher
                utility than the
             * existing best, then this will replace it
                 */
            if(combinationValue>bestCombinationValue){
```

```
                bestCombination[nodeIndex] =
                    combination;
                bestCombinationValue = combinationValue
                    ;
                agent.estimatedUtility =
                    combinationValue;
            }
        }
    }
    /* If we've got this far, then the edge has been
       approved by both
     * neighbours, which means it needs to be added,
        and the deprecated
     * links removed */
    agents[0].neighbours.add(agents[1]);
    agents[1].neighbours.add(agents[0]);
    for(int nodeIndex=0;nodeIndex<2;nodeIndex++){
        for(int neighbIndex=0;
                neighbIndex<agents[nodeIndex].
                    neighbours.numObjs;
                neighbIndex++){
            if(!bestCombination[nodeIndex]
                    .contains(agents[nodeIndex].
                        neighbours.objs[neighbIndex])){
                if(neighboursToRemove[agents[nodeIndex
                    ].index]==null){
                    neighboursToRemove[agents[nodeIndex
                        ].index]
                            = new Bag();
                }
                neighboursToRemove[agents[nodeIndex].
                    index]
                        .add(agents[nodeIndex].
                            neighbours.objs[neighbIndex
                            ]);
                if(neighboursToRemove[((Agent)agents[
                    nodeIndex]
                        .neighbours.objs[neighbIndex]).
                            index]==null){
                    neighboursToRemove[((Agent)agents[
                        nodeIndex]
                            .neighbours.objs[
                                neighbIndex]).index]
                            = new Bag();
                }
```

```
                        neighboursToRemove[((Agent)agents[
                            nodeIndex].neighbours.objs[
                            neighbIndex]).index]
                                .add(agents[nodeIndex]);
                        edgesRemoved++;
                        break;
                    }
                }
            }
        }
        for(int i=0; i<numAgents; i++) {
            Agent subject = (Agent)allNodes.objs[i];
            subject.removeNeighbours(neighboursToRemove[i]);
            /* Now that topological changes are complete, we'll
                record this
             * neighbours Bag as the oldNeighbours Bag for the
                next period */
            subject.oldNeighbours = new Bag(subject.neighbours)
               ;
        }

        newEdges.clear();
    }

    /* a recursive method to construct combinations */
    void getCombinations(Bag superSet, int lastPick, int
       remainingSpaces,
            Bag combination, Bag combinations) {
        if(remainingSpaces == 0) {
            combinations.add(combination);
            return;
        }
        int thisChoiceBound = superSet.numObjs -
           remainingSpaces + 1;
        int remainingSpacesNow = remainingSpaces - 1;
        for(int pickIndex = lastPick+1; pickIndex<
           thisChoiceBound; pickIndex++) {
            Object obj = superSet.objs[pickIndex];
            Bag combCtd = new Bag(combination);
            combCtd.add(obj);
            getCombinations(superSet, pickIndex,
               remainingSpacesNow, combCtd,
                        combinations);
        }
    }
```

```
void discoverComponents() {
    // refresh the bag of components
    components = new Bag();

    // node by node, check discovered components
       sequentially for membership.
    // if absent, it becomes the start node of a new
       component exploration
    for(int i=0; i<numAgents; i++) {
        Agent agent = (Agent)allNodes.objs[i];

        boolean componentFound = false;

        for(int componentIndex=0;
                componentIndex<components.numObjs;
                componentIndex++) {
            Bag existingComponent = (Bag)components.objs[
               componentIndex];

            if(existingComponent.contains(agent)) {
                componentFound = true;
            }
        }

        if(componentFound) continue;

        // if an existing component isn't that of the agent
           , we start a new
        // one and explore
        Bag newComponent = new Bag();
        newComponent.add(agent);
        components.add(newComponent);
        agent.component = newComponent;
        exploreComponent(agent,newComponent);
    }
}

void exploreComponent(Agent subject, Bag component) {
    Bag neighbours = subject.neighbours;

    for(int i=0; i<neighbours.numObjs; i++) {
        Agent neighbour
                = (Agent)neighbours.objs[i];
        if(!component.contains(neighbour)) {
            component.add(neighbour);
            neighbour.component = component;
```

```
                exploreComponent(neighbour,component);
            }
        }
    }

    int shortestPath(int pathLength, Object destinationNode,
       Object avoidNode,
            Bag visitedNodes, Bag oldTier) {
        pathLength++;

        Bag newTier = new Bag();
        for(int i=0; i<oldTier.numObjs; i++) {
            Agent agent = (Agent)oldTier.objs[i];
            Bag neighbours = agent.neighbours;

            // first we'll go through the neighbours From which
                the In edges come
            for(int j=0; j<neighbours.numObjs; j++) {
                Agent neighbour = (Agent)neighbours.objs[j];

                // we want to make sure that neighbour is not
                   the original node,
                // avoidNode
                // we only want to continue if we have not
                   already found a
                // faster route to that node: if it is not in a
                    lower tier
                if(!(neighbour.equals(avoidNode))
                        && (!visitedNodes.contains(neighbour)))
                            {
                    // we check whether we've reached the
                       destination
                    if(neighbour.equals(destinationNode) ==
                       true) {
                        return pathLength;
                    }
                    // if not, we extend the path so far by the
                        current node,
                    // and search on from there
                    else {
                        newTier.add(neighbour);
                        visitedNodes.add(neighbour);
                    }
                }
            }
        }
```

```
        // having built the next tier we want to pass it to a
           deeper recurrence
        // of the same algorithm, all the while keeping track
           of the number of
        // tiers
        if(!newTier.isEmpty()) {
            pathLength = shortestPath(pathLength,
               destinationNode,avoidNode,
                    visitedNodes,newTier);
            return pathLength;
        }
        else {
            // this shouldn't happen
            System.out.println("No path found to " + ((Agent)
               destinationNode).index
                    + " from " + ((Agent)avoidNode).index);
            return numAgents;
        }
    }

    /* public void updateStats()
     * checks various network statistics
     * we want this to be run only on request, to update and
        report the stats
     * for the network
     * so we want clustering distribution, degree distribution,
         centrality,
     * network clustering and support,
     * and component sizes -- possibly the component size
        distribution
     */
    public void updateStats() {
        averageDegree = 0;
        averageClustering = 0;
        averageEdgeAge = 0;
        clustering = 0;
        support = 0;
        degreeDistribution = new int[numAgents];
        clusteringDistribution = new double[numAgents];

        numRelationships = 0;
        averageProduct = .0;
        averageDiversityMeasure = 0.0;
        int maxTotalRelationships = 0;
```

```
int largestComponentSize = 0;
int secondComponentSize = 0;
int totalPotentialTriads = 0;
int completeTriads = 0;
int supportedLinks = 0;
int potentialSupport = 0;

discoverComponents();

for(int i=0; i<numAgents; i++) {
    Agent agent = (Agent)allNodes.objs[i];
    Bag agentRelationships = agent.neighbours;

    numRelationships += agentRelationships.size();
    maxTotalRelationships += maxRelationships;

    averageProduct += agent.product;

    Bag component = (Bag)agent.component;
    int componentSize = component.numObjs;
    if(componentSize>largestComponentSize) {
        secondComponentSize = largestComponentSize;
        secondComponent = largestComponent;

        largestComponentSize = componentSize;
        largestComponent = component;
    }
    else if(componentSize>secondComponentSize
            && componentSize!=largestComponentSize) {
        secondComponentSize = componentSize;
        secondComponent = component;
    }

    int degree = agent.neighbours.numObjs;
    degreeDistribution[i] = degree;
    averageDegree = averageDegree + (double)degree;

    int commonNeighbours = 0;
    int potentialTriads = 0;
    boolean[] supportFound = new boolean[
       agentRelationships.numObjs];

    for(int j=0; j<agentRelationships.numObjs; j++) {
        Agent neighbour = (Agent)agentRelationships.
            objs[j];
```

```
Bag neighbourRelationships = neighbour.
    neighbours;

// support is a link property; one and only one
     supporting node
// (edge-pair) need be found to qualify that
    edge as supported
// the way it's set up here you will count each
     link twice -
// once with each participant as the subject,
    agent
supportFound[j] = false;
potentialSupport++;

kLoop:for(int k=0; k<j; k++) {
     Agent otherNeighbour = (Agent)
        agentRelationships.objs[k];
     Bag otherNeighbourNeighbours =
        otherNeighbour.neighbours;

     // this pair of neighbours represents a
        potential triad,
     // regardless of whether they actually are
     potentialTriads++;

     for(int l=0; l<otherNeighbourNeighbours.
        numObjs; l++) {
          Agent possCommonNeighbour
                  = (Agent)
                      otherNeighbourNeighbours.
                      objs[l];
          if(neighbourRelationships.contains(
             possCommonNeighbour)) {
               if(supportFound[j]==false) {
                    supportedLinks++;
                    supportFound[j] = true;
               }
               if(supportFound[k]==false) {
                    supportedLinks++;
                    supportFound[k] = true;
               }
               // as this pair of neighbours are
                  themselves
               // neighbours, the clustering will
                  increase by 1
               commonNeighbours++;
```

```
                        break;
                    }
                }
            }
        }
        totalPotentialTriads = totalPotentialTriads +
            potentialTriads;
        if(potentialTriads>0) clusteringDistribution[i] = (
            double)commonNeighbours/(double)potentialTriads;
        averageClustering = averageClustering +
            clusteringDistribution[i];
        completeTriads = completeTriads + commonNeighbours;
    }

    averageEdgeAge = averageEdgeAge / numRelationships;
    numRelationships = numRelationships/2;

    completeness =
            (double) numRelationships /
                maxTotalRelationships;
    averageProduct = averageProduct / numAgents;
    averageDiversityMeasure = averageDiversityMeasure /
        numAgents;

    relationshipsBroken = edgesRemoved;
    edgesRemoved = 0;

    // calculate clustering and support
    if(totalPotentialTriads>0) clustering
            = (double) completeTriads / (double)
                totalPotentialTriads;
    if(potentialSupport>0) support
            = (double)supportedLinks / (double)
                potentialSupport;

    // turn sums into averages, and absolute sizes into
        population shares
    averageDegree = averageDegree/numAgents;
    averageClustering = averageClustering/numAgents;
    largestComponentShare = (double)largestComponentSize/
        numAgents;
    secondComponentShare = (double)secondComponentSize/
        numAgents;

    // if the network is connected, then we want to check
        the average
```

```
        // shortest path length
        averageShortestPath = 0.0;
        for(int i=0; i<largestComponentSize; i++) {
            Agent startAgent = (Agent)largestComponent.objs[i];
            for(int j=0; j<i; j++) {
                Agent finishAgent = (Agent)largestComponent.
                    objs[j];

                Bag oldTier = new Bag();
                oldTier.add(startAgent);
                averageShortestPath += shortestPath(0,
                    finishAgent,
                        startAgent,new Bag(oldTier),oldTier);
            }
        }
        averageShortestPath = averageShortestPath * 2.0 / (
            numAgents*(numAgents-1));
    }
}
```

5.A.3 Model with only random neighbours

```
/*
 Copyright 2011 by Tom Wilkinson and Cardiff University
 Licensed under the Academic Free License version 3.0
 See the file "LICENSE" for more information
*/
package anarchyrandom;
import util.*;

/* Anarchy Trade model allows agents to predict the effects of
    adding neighbours
* on those neighbours' diversities!
*/
public class RandomNetwork {
    Bag allNodes;
    MersenneTwisterFast random;

    final boolean localSearch;

    final int numAgents;
    final int maxRelationships;
    final int exchangeRate;
    final double globalSearchProb;

    final double depreciation;
```

```
final double innovation;

final double invNumAgents;

Bag newEdges;

int edgesRemoved;
boolean statsTaken;
Bag components;
Bag largestComponent;
Bag secondComponent;

// instantaneous network statistics:
double averageDegree;
int[] degreeDistribution;
double averageClustering;
double[] clusteringDistribution;
double clustering;
double support;
double[] centrality;
double largestComponentShare;
double secondComponentShare;
double averageShortestPath;
double averageEdgeAge;

Bag parents;

// dynamic network statistics:
double averageProduct;
double averageDiversityMeasure;
int numRelationships;
double completeness;
int relationshipsBroken;

public RandomNetwork(boolean localSearch, int numAgents,
   int maxRelationships,
        int exchangeRate, double globalSearchProb, double
           depreciation,
        double innovation) {
    this.localSearch = localSearch;

    this.numAgents = numAgents;
    this.maxRelationships = maxRelationships;
    this.exchangeRate = exchangeRate;
    this.globalSearchProb = globalSearchProb;
    this.depreciation = depreciation;
```

```
        this.innovation = innovation;

        invNumAgents = 1.0/(double)numAgents;

        newEdges = new Bag();

        random = new MersenneTwisterFast();
    }

    // Transfers are now called in from the various agents, and
        then redistributed
    public void exchangeTransfers() {
    }


    /* After transfers but before choices, this provides new
       neighbours to those
     * agents that can sustain them
     */
    public void neighbourSearch() {
        Bag classifieds = new Bag(allNodes);

        /* local search should happen first, to reduce the
           possibility
         * that there are no viable local search choices left
            in classifieds:
         * - if there is a clique there could still be no local
             search choices */

        Agent agent =
                (Agent)classifieds.objs[random.nextInt(
                    classifieds.numObjs)];

        // we create a new bag to keep track of neighbours we'
           ve visited
        // in local search
        Bag neighbours = new Bag(agent.neighbours);

        // now a specific classifieds bag stopping us selecting
        // inappropriate new links for this particular agent
        Bag tempClassifieds = new Bag(classifieds);
        tempClassifieds.remove(agent);
        tempClassifieds.removeAll(neighbours);
        if(tempClassifieds.isEmpty()) {
            classifieds.remove(agent);
        }
```

```
// with a given probability a new neighbour is found by
    global search
if(random.nextBoolean(globalSearchProb)) {
    int potentialClassified =
            random.nextInt(tempClassifieds.numObjs);
    Agent potentialNeighbour = (Agent)tempClassifieds
            .objs[potentialClassified];
    /* now we add these agents as a pair to the bag of
       new edges
     * to check out */
    newEdges.add(new Agent[]{agent,potentialNeighbour})
        ;

    classifieds.remove(agent);
    classifieds.remove(potentialNeighbour);
}
// otherwise local search is used - may not be
   successful!
else if(localSearch) {
    // first we randomly choose a neighbour from whom
       to pick a
    // common neighbour
    while(!neighbours.isEmpty()) {
        int neighbourIndex = random.nextInt(neighbours.
           numObjs);
        Agent viaNeighbour = (Agent)neighbours.objs[
           neighbourIndex];

        Bag potentials = new Bag(viaNeighbour.
           neighbours);
        potentials.remove(agent);

        // now we randomly search through that agent's
        // neighbours
        while(!potentials.isEmpty()) {
            int potentialIndex = random.nextInt(
               potentials.numObjs);
            Agent potential = (Agent)potentials.objs[
               potentialIndex];

            if(tempClassifieds.contains(potential)) {
                newEdges.add(new Agent[]{agent,
                    potential});

                classifieds.remove(agent);
```

```
                    classifieds.remove(potential);
                    return;
                }
                else potentials.remove(potential);
            }
            neighbours.remove(viaNeighbour);
        }
    }
}

/* Agents evaluate relationships, they are then ranked, and
    this is followed
 * by the actual change to network topology
 */
public void neighbourChoose() {
}

void discoverComponents() {
    // refresh the bag of components
    components = new Bag();

    // node by node, check discovered components
       sequentially for membership.
    // if absent, it becomes the start node of a new
       component exploration
    for(int i=0; i<numAgents; i++) {
        Agent agent = (Agent)allNodes.objs[i];

        boolean componentFound = false;

        for(int componentIndex=0;
                componentIndex<components.numObjs;
                componentIndex++) {
            Bag existingComponent = (Bag)components.objs[
               componentIndex];

            if(existingComponent.contains(agent)) {
                componentFound = true;
            }
        }

        if(componentFound) continue;

        // if an existing component isn't that of the agent
           , we start a new
        // one and explore
```

```
            Bag newComponent = new Bag();
            newComponent.add(agent);
            components.add(newComponent);
            agent.component = newComponent;
            exploreComponent(agent,newComponent);
        }
    }

    void exploreComponent(Agent subject, Bag component) {
        Bag neighbours = subject.neighbours;

        for(int i=0; i<neighbours.numObjs; i++) {
            Agent neighbour
                    = (Agent)neighbours.objs[i];
            if(!component.contains(neighbour)) {
                component.add(neighbour);
                neighbour.component = component;
                exploreComponent(neighbour,component);
            }
        }
    }

    int shortestPath(int pathLength, Object destinationNode,
       Object avoidNode,
            Bag visitedNodes, Bag oldTier) {
        pathLength++;

        Bag newTier = new Bag();
        for(int i=0; i<oldTier.numObjs; i++) {
            Agent agent = (Agent)oldTier.objs[i];
            Bag neighbours = agent.neighbours;

            // first we'll go through the neighbours From which
                the In edges come
            for(int j=0; j<neighbours.numObjs; j++) {
                Agent neighbour = (Agent)neighbours.objs[j];

                // we want to make sure that neighbour is not
                    the original node,
                // avoidNode
                // we only want to continue if we have not
                    already found a
                // faster route to that node: if it is not in a
                     lower tier
                if(!(neighbour.equals(avoidNode))
```

```
                    && (!visitedNodes.contains(neighbour)))
                        {
                    // we check whether we've reached the
                       destination
                    if(neighbour.equals(destinationNode) ==
                       true) {
                        return pathLength;
                    }
                    // if not, we extend the path so far by the
                        current node,
                    // and search on from there
                    else {
                        newTier.add(neighbour);
                        visitedNodes.add(neighbour);
                    }
                }
            }
        }

        // having built the next tier we want to pass it to a
           deeper recurrence
        // of the same algorithm, all the while keeping track
           of the number of
        // tiers
        if(!newTier.isEmpty()) {
            pathLength = shortestPath(pathLength,
               destinationNode,avoidNode,
                    visitedNodes,newTier);
            return pathLength;
        }
        else {
            // this shouldn't happen
            System.out.println("No path found to " + ((Agent)
               destinationNode).index
                    + " from " + ((Agent)avoidNode).index);
            return numAgents;
        }
    }

    /* public void updateStats()
     * checks various network statistics
     * we want this to be run only on request, to update and
        report the stats
     * for the network
     * so we want clustering distribution, degree distribution,
        centrality,
```

```
     * network clustering and support,
     * and component sizes - possibly the component size
        distribution
     */
    public void updateStats() {
        averageDegree = 0;
        averageClustering = 0;
        averageEdgeAge = 0;
        clustering = 0;
        support = 0;
        degreeDistribution = new int[numAgents];
        clusteringDistribution = new double[numAgents];

        numRelationships = 0;
        averageProduct = .0;
        averageDiversityMeasure = 0.0;
        int maxTotalRelationships = 0;

        int largestComponentSize = 0;
        int secondComponentSize = 0;
        int totalPotentialTriads = 0;
        int completeTriads = 0;
        int supportedLinks = 0;
        int potentialSupport = 0;

        discoverComponents();

        for(int i=0; i<numAgents; i++) {
            Agent agent = (Agent)allNodes.objs[i];
            Bag agentRelationships = agent.neighbours;

            numRelationships += agentRelationships.size();
            maxTotalRelationships += maxRelationships;

            averageProduct += agent.product;

            Bag component = (Bag)agent.component;
            int componentSize = component.numObjs;
            if(componentSize>largestComponentSize) {
                secondComponentSize = largestComponentSize;
                secondComponent = largestComponent;

                largestComponentSize = componentSize;
                largestComponent = component;
            }
            else if(componentSize>secondComponentSize
```

```
            && componentSize!=largestComponentSize) {
        secondComponentSize = componentSize;
        secondComponent = component;
    }

    int degree = agent.neighbours.numObjs;
    degreeDistribution[i] = degree;
    averageDegree = averageDegree + (double)degree;

    int commonNeighbours = 0;
    int potentialTriads = 0;
    boolean[] supportFound = new boolean[
       agentRelationships.numObjs];

    for(int j=0; j<agentRelationships.numObjs; j++) {
        Agent neighbour = (Agent)agentRelationships.
           objs[j];
        Bag neighbourRelationships = neighbour.
           neighbours;

        // support is a link property; one and only one
            supporting node
        // (edge-pair) need be found to qualify that
           edge as supported
        // the way it's set up here you will count each
            link twice -
        // once with each participant as the subject,
           agent
        supportFound[j] = false;
        potentialSupport++;

        kLoop:for(int k=0; k<j; k++) {
            Agent otherNeighbour = (Agent)
               agentRelationships.objs[k];
            Bag otherNeighbourNeighbours =
               otherNeighbour.neighbours;

            // this pair of neighbours represents a
               potential triad,
            // regardless of whether they actually are
            potentialTriads++;

            for(int l=0; l<otherNeighbourNeighbours.
               numObjs; l++) {
                Agent possCommonNeighbour
```

```
                                = (Agent)
                                    otherNeighbourNeighbours.
                                    objs[1];
                            if(neighbourRelationships.contains(
                                possCommonNeighbour)) {
                                if(supportFound[j]==false) {
                                    supportedLinks++;
                                    supportFound[j] = true;
                                }
                                if(supportFound[k]==false) {
                                    supportedLinks++;
                                    supportFound[k] = true;
                                }
                                // as this pair of neighbours are
                                    themselves
                                // neighbours, the clustering will
                                    increase by 1
                                commonNeighbours++;
                                break;
                            }
                        }
                    }
                }
                totalPotentialTriads = totalPotentialTriads +
                    potentialTriads;
                if(potentialTriads>0) clusteringDistribution[i] = (
                    double)commonNeighbours/(double)potentialTriads;
                averageClustering = averageClustering +
                    clusteringDistribution[i];
                completeTriads = completeTriads + commonNeighbours;
            }

            averageEdgeAge = averageEdgeAge / numRelationships;
            numRelationships = numRelationships/2;

            completeness =
                    (double) numRelationships /
                        maxTotalRelationships;
            averageProduct = averageProduct / numAgents;
            averageDiversityMeasure = averageDiversityMeasure /
                numAgents;

            relationshipsBroken = edgesRemoved;
            edgesRemoved = 0;

            // calculate clustering and support
```

```
        if(totalPotentialTriads>0) clustering
                = (double) completeTriads / (double)
                    totalPotentialTriads;
        if(potentialSupport>0) support
                = (double)supportedLinks / (double)
                    potentialSupport;

        // turn sums into averages, and absolute sizes into
           population shares
        averageDegree = averageDegree/numAgents;
        averageClustering = averageClustering/numAgents;
        largestComponentShare = (double)largestComponentSize/
           numAgents;
        secondComponentShare = (double)secondComponentSize/
           numAgents;

        // if the network is connected, then we want to check
           the average
        // shortest path length
        averageShortestPath = 0.0;
        for(int i=0; i<largestComponentSize; i++) {
            Agent startAgent = (Agent)largestComponent.objs[i];
            for(int j=0; j<i; j++) {
                Agent finishAgent = (Agent)largestComponent.
                    objs[j];

                Bag oldTier = new Bag();
                oldTier.add(startAgent);
                averageShortestPath += shortestPath(0,
                    finishAgent,
                        startAgent,new Bag(oldTier),oldTier);
            }
        }
        averageShortestPath = averageShortestPath * 2.0 / (
           numAgents*(numAgents-1));
    }
}
```

License

GNU Free Documentation License

Version 1.3, 3 November 2008

Copyright © 2000, 2001, 2002, 2007, 2008 Free Software Foundation, Inc. <http://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

0. PREAMBLE The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A “Secondary Section” is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document’s overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The “Invariant Sections” are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The “Cover Texts” are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A “Transparent” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not “Transparent” is called “Opaque”.

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The “Title Page” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, “Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

The “publisher” means any person or entity that distributes copies of the Document to the public.

A section “Entitled XYZ” means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as “Acknowledgements”, “Dedications”, “Endorsements”, or “History”.) To “Preserve the Title” of such

a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under

precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission. B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement. C. State on the Title page the name of the publisher of the Modified Version, as the publisher. D. Preserve all the copyright notices of the Document. E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices. F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below. G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice. H. Include an unaltered copy of this License. I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence. J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission. K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein. L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles. M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version. N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section. O. Preserve any Warranty Disclaimers. If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the

same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections

with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING "Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of

such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.