Accelerating ADMM for Efficient Simulation and Optimization

JUYONG ZHANG*†, University of Science and Technology of China
YUE PENG*, University of Science and Technology of China
WENQING OUYANG*, University of Science and Technology of China
BAILIN DENG, Cardiff University

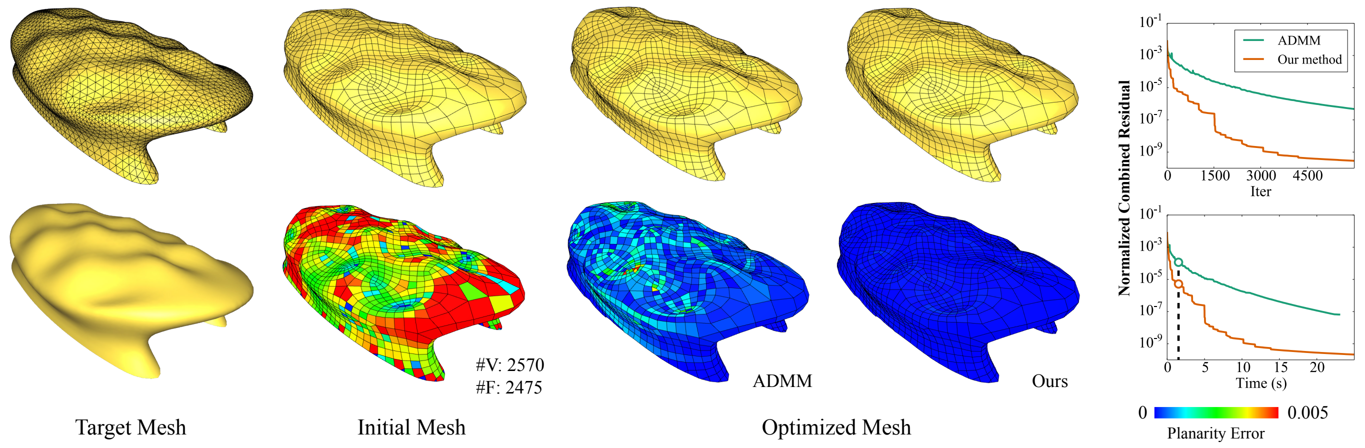

Fig. 1. We apply our accelerated ADMM solver to optimize a quad mesh, subject to hard constraints of face planarity and soft constraints of closeness to a reference surface. Our solver leads to a faster decrease of combined residual than the original ADMM, achieving better satisfaction of hard constraints within the same computational time (highlighted in the plot in bottom right).

The alternating direction method of multipliers (ADMM) is a popular approach for solving optimization problems that are potentially non-smooth and with hard constraints. It has been applied to various computer graphics applications, including physical simulation, geometry processing, and image processing. However, ADMM can take a long time to converge to a solution of high accuracy. Moreover, many computer graphics tasks involve non-convex optimization, and there is often no convergence guarantee for ADMM on such problems since it was originally designed for convex optimization. In this paper, we propose a method to speed up ADMM using Anderson acceleration, an established technique for accelerating fixed-point iterations. We show that in the general case, ADMM is a fixed-point iteration of the second primal variable and the dual variable, and Anderson acceleration can be directly applied. Additionally, when the problem has a separable target function and satisfies certain conditions, ADMM becomes a fixed-point iteration of only one variable, which further reduces the computational overhead of Anderson acceleration. Moreover, we analyze a particular non-convex problem structure that is common in computer graphics, and prove the convergence of ADMM on such problems under mild assumptions. We apply our acceleration technique on a variety of optimization problems in computer graphics, with notable improvement on their convergence speed.

*Equal contributions.
†Corresponding author (juyong@ustc.edu.cn).

Authors' addresses: {Juyong Zhang, Yue Peng, Wenqing Ouyang}, University of Science and Technology of China, 96 Jinzhai Road, Hefei 230026, Anhui, China, {juyong@ustc.edu.cn, echoyue@mail.ustc.edu.cn, wq8809@mail.ustc.edu.cn}; Bailin Deng, Cardiff University, 5 The Parade, Cardiff CF24 3AA, Wales, United Kingdom, DengB3@cardiff.ac.uk.

CCS Concepts: • **Computing methodologies** → **Computer graphics**; **Animation**; • **Theory of computation** → *Nonconvex optimization*;

Additional Key Words and Phrases: Physics Simulation, Geometry Optimization, ADMM, Anderson Acceleration

ACM Reference Format:
Juyong Zhang, Yue Peng, Wenqing Ouyang, and Bailin Deng. 2019. Accelerating ADMM for Efficient Simulation and Optimization. *ACM Trans. Graph.* 38, 6, Article 163 (November 2019), 21 pages. https://doi.org/10.1145/3355089.3356491

1 INTRODUCTION

Many tasks in computer graphics involve solving optimization problems. For example, a geometry processing task may compute the vertex positions of a deformed mesh by minimizing its deformation energy [Sorkine and Alexa 2007], whereas a physical simulation task may optimize the node positions of a system to enforce physics laws that govern its behavior [Martin et al. 2011; Schumacher et al. 2012]. Such tasks are often formulated as *unconstrained* optimization, where the target function penalizes the violation of certain conditions so that they are satisfied as much as possible by the solution. It has been an active research topic to develop fast numerical solvers for such problems, with various methods proposed in the past [Sorkine and Alexa 2007; Liu et al. 2008; Bouaziz et al. 2012; Liu et al. 2013; Bouaziz et al. 2014; Wang 2015; Kovalsky et al. 2016; Liu et al. 2017; Shtengel et al. 2017; Rabinovich et al. 2017].

On the other hand, some applications involve optimization with *hard constraints*, i.e., conditions that need to be enforced strictly. Such *constrained* optimization problems are often more difficult to solve [Nocedal and Wright 2006]. One possible solution strategy is to introduce a quadratic penalty term for the hard constraints with a large weight, thereby converting it into an unconstrained problem that is easier to handle. However, to strictly enforce the hard constraints, their penalty weight needs to approach infinity [Nocedal and Wright 2006], which can cause instability for numerical solvers. More sophisticated techniques, such as sequential quadratic programming or the interior-point method, can enforce constraints without stability issues. However, these solvers often incur high computational costs and may not meet the performance requirements for graphics applications. It becomes even more challenging for non-smooth problems where the target function is not everywhere differentiable, as many constrained optimization solvers require gradient information and may not be applicable for such cases.

In recent years, the alternating direction method of multipliers (ADMM) [Boyd et al. 2011] has become a popular approach for solving optimization problems that are potentially non-smooth and with hard constraints. The key idea is to introduce auxiliary variables and derive an equivalent problem with a separable target function, subject to a linear compatibility constraint between the original variables and the auxiliary variables [Combettes and Pesquet 2011]. ADMM searches for a solution to this converted problem by alternately updating the original variables, the auxiliary variables, and the dual variables. With properly chosen auxiliary variables, each update step can reduce to simple sub-problems that can be solved efficiently, often in parallel with closed-form solutions. In addition, ADMM does not rely on the smoothness of the problem, and converges quickly to a solution of moderate accuracy [Boyd et al. 2011]. Such properties make ADMM an attractive choice for solving large-scale optimization problems in various applications such as signal processing [Chartrand and Wohlberg 2013; Simonetto and Leus 2014], image processing [Figueiredo and Bioucas-Dias 2010; Almeida and Figueiredo 2013], and computer vision [Liu et al. 2013]. Recently, ADMM has also been applied for computer graphics problems such as geometry processing [Bouaziz et al. 2013; Neumann et al. 2013; Zhang et al. 2014; Xiong et al. 2014; Neumann et al. 2014], physics simulation [Gregson et al. 2014; Pan and Manocha 2017; Overby et al. 2017], and computational photography [Heide et al. 2016; Xiong et al. 2017; Wang et al. 2018].

Despite the effectiveness and versatility of ADMM, there are still two major limitations for its use in computer graphics. First, although ADMM converges quickly in initial iterations, its final convergence might be slow [Boyd et al. 2011]. This makes it impractical for problems with a strong demand for solution accuracy, such as those with strict requirements on the satisfaction of hard constraints. Recent attempts to accelerate ADMM such as [Goldstein et al. 2014; Kadkhodaie et al. 2015; Zhang and White 2018] are only designed for convex problems, which limits their applications in computer graphics. Second, ADMM was originally designed for convex problems, whereas many computer graphics tasks involve non-convex optimization. Although ADMM turns out to be effective for many non-convex problems in practice, its convergence for general non-convex optimization remains an open research question. Recent convergence results such as [Li and Pong 2015; Hong et al. 2016; Magnússon et al. 2016; Wang et al. 2019] rely on strong assumptions that are not satisfied by many computer graphics problems.

This paper addresses these two issues of ADMM. First, we propose a method to accelerate ADMM for non-convex optimization problems. Our approach is based on Anderson acceleration [Anderson 1965; Walker and Ni 2011], a well-established technique for accelerating fixed-point iterations. Previously, Anderson acceleration has been applied to local-global solvers for unconstrained optimization problems in computer graphics [Peng et al. 2018]. Our approach expands its applicability to many constrained optimization problems as well as other unconstrained problems where local-solver solvers are not feasible. To this end, we need to solve two problems: (i) we must find a way to interpret ADMM as a fixed-point iteration; (ii) as Anderson acceleration can become unstable, we should define criteria to accept the accelerated iterate and a fall-back strategy when it is not accepted, similar to [Peng et al. 2018]. We show that in the general case ADMM is a fixed-point iteration of the second primal variable and the dual variable, and we can evaluate the effectiveness of an accelerated iterate via its *combined residual* which is known to vanish when the solver converges. Moreover, when the problem structure satisfies some mild conditions, one of these two variables can be determined from the other one; in this case ADMM becomes a fixed-point iteration of only one variable with less computational overhead, and we can accept an accelerated iterate based on a more simple condition. We apply this method to a variety of ADMM solvers for computer graphics problems, and observe a notable improvement in their convergence rates.

Additionally, we provide a new convergence proof of ADMM on non-convex problems, under weaker assumptions than the convergence results in [Li and Pong 2015; Hong et al. 2016; Magnússon et al. 2016; Wang et al. 2019]. For a particular problem structure that is common in computer graphics, we also provide sufficient conditions for the global linear convergence of ADMM. Our proofs shed new light on the convergence properties of non-convex ADMM solvers.

2 RELATED WORK

Optimization solvers in computer graphics. The development of efficient optimization solvers has been an active research topic in computer graphics. One particular type of method, called local-global solvers, has been widely used for unconstrained optimization in geometry processing and physical simulation. For geometry processing, Sorkine and Alexa [2007] proposed a local-global approach to minimize deformation energy for as-rigid-as-possible mesh surface modeling. Liu et al. [2008] developed a similar method to perform conformal and isometric parameterization for triangle meshes. Bouaziz et al. [2012] extended the approach to a unified framework for optimizing discrete shapes. For physical simulation, Liu et al. [2013] proposed a local-global solver for optimization-based simulation of mass-spring systems. Bouaziz et al. [2014] extended this approach to the projective dynamics framework for implicit time integration of physical systems via energy minimization.

Local-global solvers often converge quickly to an approximate solution, but may be slow for final convergence. Other methods have been proposed to achieve improved convergence rates. For geometry processing, Kovalsky et al. [2016] achieved a fast convergence of geometric optimization by iteratively minimizing a local quadratic proxy function. Rabinovich et.al. [2017] proposed a scalable approach to compute locally injective mappings, via local-global minimization of a reweighted proxy function. Claici et al. [2017] proposed a preconditioner for fast minimization of distortion energies. Shtengel et al. [2017] applied the idea of majorization-minimization [Lange 2004] to iteratively update and minimize a convex majorizer of the target energy in geometric optimization. Zhu et al. [2018] proposed a fast solver for distortion energy minimization, using a blended quadratic energy proxy together with improved line-search strategy and termination criteria. For physical simulation, Wang [2015] proposed a Chebyshev semi-iterative acceleration technique for projective dynamics. Later, Wang and Yang [2016] developed a GPU-friendly gradient descent method for elastic body simulation, using Jacobi preconditioning and Chebyshev acceleration. Liu et al. [2017] proposed an L-BFGS solver for physical simulation, with faster convergence than the projective dynamics solver from [Bouaziz et al. 2014]. Brandt et al. [2018] performed projective dynamics simulation in a reduced subspace, to compute fast approximate solutions for high-resolution meshes.

ADMM. ADMM is a popular solver for optimization problems with separable target functions and linear side constraints [Boyd et al. 2011]. Using auxiliary variables and indicator functions, such formulation allows for non-smooth optimization with hard constraints, with wide applications in signal processing [Erseghe et al. 2011; Simonetto and Leus 2014; Shi et al. 2014], image processing [Figueiredo and Bioucas-Dias 2010; Almeida and Figueiredo 2013], computer vision [Hu et al. 2013; Liu et al. 2013; Yang et al. 2017], computational imaging [Chan et al. 2017], automatic control [Lin et al. 2013], and machine learning [Zhang and Kwok 2014; Hajinezhad et al. 2016]. ADMM has also been used in computer graphics to handle non-smooth optimization problems [Bouaziz et al. 2013; Neumann et al. 2013; Zhang et al. 2014; Xiong et al. 2014; Neumann et al. 2014] or to benefit from its fast initial convergence [Gregson et al. 2014; Heide et al. 2016; Xiong et al. 2017; Pan and Manocha 2017; Overby et al. 2017; Wang et al. 2018].

ADMM was originally designed for convex optimization [Gabay and Mercier 1976; Fortin and Glowinski 1983; Eckstein and Bertsekas 1992]. For such problems, its global linear convergence has been established in [Lin et al. 2015; Deng and Yin 2016; Giselsson and Boyd 2017], but these proofs require both terms in the target function to be convex. In comparison, our proof of global linear convergence allows for non-convex terms in the target function, which is better aligned with computer graphics problems. In practice, ADMM works well for many non-convex problems as well [Wen et al. 2012; Chartrand 2012; Chartrand and Wohlberg 2013; Miksik et al. 2014; Lai and Osher 2014; Liavas and Sidiropoulos 2015], but it is more challenging to establish its convergence for general non-convex problems. Only very recently have such convergence proofs been given under strong assumptions [Li and Pong 2015; Hong et al. 2016; Magnússon et al. 2016; Wang et al. 2019]. We provide in this paper a general proof of convergence for non-convex problems under weaker assumptions.

It is well known that ADMM converges quickly to an approximate solution, but may take a long time to convergence to a solution of high accuracy [Boyd et al. 2011]. This has motivated researchers to explore acceleration techniques for ADMM. Goldstein et al. [2014] and Kadkhodaie et al. [2015] applied Nesterov's acceleration [Nesterov 1983], whereas Zhang and White [2018] applied GMRES acceleration to a special class of problems where the ADMM iterates become linear. All these methods are designed for convex problems only, which limits their applicability in computer graphics.

Anderson acceleration. Anderson acceleration [Walker and Ni 2011] is an established technique to speed up the convergence of a fixed-point iteration. It was first proposed in [Anderson 1965] for solving nonlinear integral equations, and independently rediscovered later by Pulay [1980; 1982] for accelerating the self-consistent field method in quantum chemistry. Its key idea is to utilize the m previous iterates to compute a new iterate that converges faster to the fixed point. It is indeed a quasi-Newton method for finding a root of the residual function, by approximating its inverse Jacobian using previous iterates [Eyert 1996; Fang and Saad 2009; Rohwedder and Schneider 2011]. Recently, a renewed interest in this method has led to the analysis of its convergence [Toth and Kelley 2015; Toth et al. 2017], as well as its application in various numerical problems [Sterck 2012; Lipnikov et al. 2013; Pratapa et al. 2016; Suryanarayana et al. 2019; Ho et al. 2017]. Peng et al. [Peng et al. 2018] noted that local-global solvers in computer graphics can be treated as fixed-point iteration, and applied Anderson acceleration to improve their convergence. Additionally, to address the stability issue of classical Anderson acceleration [Walker and Ni 2011; Potra and Engler 2013], they utilize the monotonic energy decrease of local-global solvers and only accept an accelerated iterate when it decreases the target energy. Fang and Saad [2009] called classical Anderson acceleration the Type-II method in an Anderson family of multi-secant methods. Another member of the family, called the type-I method, uses quasi-Newton to approximate the Jacobian of the fixed-point residual function instead [Walker and Ni 2011], and has been analyzed recently in [Zhang et al. 2018]. In this paper, we focus our discussion on the type-II method.

3 OUR METHOD

3.1 Preliminary

ADMM. Let us consider an optimization problem

$$\min_{\mathbf{x}} \ \Phi(\mathbf{x}, \mathbf{D}\mathbf{x} + \mathbf{h}). \tag{1}$$

Here $\mathbf{x}$ can be the vertex positions of a discrete geometric shape, or the node positions of a physical system at a particular time instance. The quantity $\mathbf{D}\mathbf{x} + \mathbf{h}$ encodes a transformation of the positions $\mathbf{x}$ relevant for the optimization problem, such as the deformation gradient of each tetrahedron element in an elastic object. The notation $\Phi(\mathbf{x}, \mathbf{D}\mathbf{x} + \mathbf{h})$ signifies that the target function contains a term that directly depends on $\mathbf{D}\mathbf{x} + \mathbf{h}$, such as elastic energy dependent on the deformation gradient. In some applications, the optimization enforces *hard constraints* on $\mathbf{x}$ or $\mathbf{D}\mathbf{x} + \mathbf{h}$, i.e., conditions that need to be strictly satisfied by the solution. Such hard constraints can be encoded using an *indicator function* term within the target function. Specifically, suppose we want to enforce a condition $\mathbf{y} \in C$ where $\mathbf{y}$ is a subset from the components of $\mathbf{x}$ or $\mathbf{D}\mathbf{x} + \mathbf{h}$, and C is the *feasible* *set*. Then we include the following term into Φ:

$$\sigma_C(\mathbf{y}) = \begin{cases} 0 & \text{if } \mathbf{y} \in C \\ +\infty & \text{otherwise} \end{cases}.$$

By definition, if $\mathbf{x}^*$ is a solution, then the corresponding components $\mathbf{y}^*$ must satisfy $\mathbf{y}^* \in C$; otherwise it will result in a target function value $+\infty$ instead of the minimum. Examples of such an approach to modeling hard constraints can be found in [Deng et al. 2015].

In many applications, the optimization problem (1) can be nonlinear, non-convex, and potentially non-smooth. It is challenging to solve such a problem numerically, especially when hard constraints are involved. One common technique is to introduce an auxiliary variable $\mathbf{z} = \mathbf{D}\mathbf{x} + \mathbf{h}$ to derive an equivalent problem

$$\min_{\mathbf{x},\mathbf{z}} \ \Phi(\mathbf{x},\mathbf{z}) \quad \text{s.t. } \mathbf{W}(\mathbf{z} - \mathbf{D}\mathbf{x} - \mathbf{h}) = 0, \tag{2}$$

where $\mathbf{W}$ is a diagonal matrix with positive diagonal elements. $\mathbf{W}$ can be the identity matrix in the trivial case, or a diagonal scaling matrix that improves conditioning [Giselsson and Boyd 2017; Overby et al. 2017]. ADMM [Boyd et al. 2011] is widely used to solve such problems. For ease of discussion, let us consider the problem

$$\min_{\mathbf{x},\mathbf{z}} \ \Phi(\mathbf{x},\mathbf{z}) \quad \text{s.t. } \mathbf{A}\mathbf{x} - \mathbf{B}\mathbf{z} = \mathbf{c}, \tag{3}$$

Its solution corresponds to a stationary point of the augmented Lagrangian function

$$\begin{aligned} L(\mathbf{x},\mathbf{z},\mathbf{u}) &= \Phi(\mathbf{x},\mathbf{z}) + \langle \mu\mathbf{u}, \mathbf{A}\mathbf{x} - \mathbf{B}\mathbf{z} - \mathbf{c}\rangle + \frac{\mu}{2}\|\mathbf{A}\mathbf{x} - \mathbf{B}\mathbf{z} - \mathbf{c}\|^2 \\ &= \Phi(\mathbf{x},\mathbf{z}) + \frac{\mu}{2}\|\mathbf{A}\mathbf{x} - \mathbf{B}\mathbf{z} + \mathbf{u} - \mathbf{c}\|^2 - \frac{\mu}{2}\|\mathbf{u}\|^2. \end{aligned} \tag{4}$$

Here $\mathbf{u}$ is the *dual variable* and $\mu > 0$ is the penalty parameter. Following [Boyd et al. 2011], we also call $\mathbf{x}$ and $\mathbf{z}$ the *primal variables*. ADMM searches for a stationary point by alternately updating $\mathbf{x}$, $\mathbf{z}$ and $\mathbf{u}$, resulting in the following iteration scheme [Boyd et al. 2011]:

$$\begin{aligned} \mathbf{x}^{k+1} &= \operatorname*{argmin}_{\mathbf{x}} \ L(\mathbf{x}, \mathbf{z}^k, \mathbf{u}^k), \\ \mathbf{z}^{k+1} &= \operatorname*{argmin}_{\mathbf{z}} \ L(\mathbf{x}^{k+1}, \mathbf{z}, \mathbf{u}^k), \\ \mathbf{u}^{k+1} &= \mathbf{u}^k + \mathbf{A}\mathbf{x}^{k+1} - \mathbf{B}\mathbf{z}^{k+1} - \mathbf{c}. \end{aligned} \tag{5}$$

We can also update $\mathbf{z}$ before $\mathbf{x}$, resulting in an alternative scheme:

$$\begin{aligned} \mathbf{z}^{k+1} &= \operatorname*{argmin}_{\mathbf{z}} \ L(\mathbf{x}^k, \mathbf{z}, \mathbf{u}^k), \\ \mathbf{x}^{k+1} &= \operatorname*{argmin}_{\mathbf{x}} \ L(\mathbf{x}, \mathbf{z}^{k+1}, \mathbf{u}^k), \\ \mathbf{u}^{k+1} &= \mathbf{u}^k + \mathbf{A}\mathbf{x}^{k+1} - \mathbf{B}\mathbf{z}^{k+1} - \mathbf{c}. \end{aligned} \tag{6}$$

In this paper, we refer to the scheme (5) as **x-z-u** iteration, and the scheme (6) as **z-x-u** iteration. In both cases, the updates for $\mathbf{z}$ and $\mathbf{x}$ often reduce to simple subproblems that can potentially be solved in parallel. According to [Boyd et al. 2011], the optimality condition of ADMM is that both its *primal residual* and *dual residual* vanish. For both iteration schemes above, the primal residual is defined as

$$\mathbf{r}_{\mathrm{p}}^{k+1} = \mathbf{A}\mathbf{x}^{k+1} - \mathbf{B}\mathbf{z}^{k+1} - \mathbf{c}.$$

As for the dual residual: for the **x-z-u** iteration it is defined as

$$\mathbf{r}_{\mathrm{d}}^{k+1} = \mu\mathbf{A}^T\mathbf{B}(\mathbf{z}^{k+1} - \mathbf{z}^k), \tag{7}$$

whereas for the **z-x-u** iteration it is defined as

$$\mathbf{r}_{\mathrm{d}}^{k+1} = \mu\mathbf{B}^T\mathbf{A}(\mathbf{x}^{k+1} - \mathbf{x}^k). \tag{8}$$

Intuitively, the primal residual measures the violation of the linear side constraint, whereas the dual residual measures the violation of the dual feasibility condition [Boyd et al. 2011]. Accordingly, ADMM is terminated when both $\|\mathbf{r}_{\mathrm{p}}^{k+1}\|$ and $\|\mathbf{r}_{\mathrm{d}}^{k+1}\|$ are small enough.

Anderson acceleration. ADMM is easy to parallelize and convergences quickly to an approximate solution. However, it can take a long time to converge to a solution of high accuracy [Boyd et al. 2011]. In the following subsections, we will discuss how to apply Anderson acceleration [Walker and Ni 2011] to improve its convergence. Anderson acceleration is a technique to speed up the convergence of a fixed-point iteration $G : \mathbb{R}^n \mapsto \mathbb{R}^n$, by utilizing the current iterate as well as m previous iterates. Let $\mathbf{q}^{k-m}, \mathbf{q}^{k-m+1}, \ldots, \mathbf{q}^k$ be the latest $m+1$ iterates, and denote their residuals under mapping G as $F^{k-m}, F^{k-m+1}, \ldots, F^k$, where $F^j = G(\mathbf{q}^j) - \mathbf{q}^j$ $(j = k-m, \ldots, k)$. Then the accelerated iterate is computed as

$$\begin{aligned} \mathbf{q}_{\mathrm{AA}}^{k+1} = \ & (1-\beta)\left(\mathbf{q}^k - \sum_{j=1}^m \theta_j^*(\mathbf{q}^{k-j+1} - \mathbf{q}^{k-j})\right) \\ & + \beta\left(G(\mathbf{q}^k) - \sum_{j=1}^m \theta_j^*(G(\mathbf{q}^{k-j+1}) - G(\mathbf{q}^{k-j}))\right), \end{aligned} \tag{9}$$

where $(\theta_1^*, \ldots, \theta_m^*)$ is the solution to a linear least-squares problem:

$$\min_{(\theta_1,\ldots,\theta_m)} \left\| F^k - \sum_{j=1}^m \theta_j (F^{k-j+1} - F^{k-j}) \right\|^2. \tag{10}$$

In Eq. (9), $\beta \in (0, 1]$ is a mixing parameter, and is typically set to 1 [Walker and Ni 2011]. We follow this convention throughout this paper. Previously, Anderson acceleration has been applied to speed up local-global solvers in computer graphics [Peng et al. 2018].

3.2 Anderson acceleration of ADMM: the general approach

To speed up ADMM with Anderson acceleration, we must first define its iteration scheme as a fixed-point iteration. For the **x-z-u** iteration, we note that $\mathbf{x}^{k+1}$ is dependent only on $\mathbf{z}^k$ and $\mathbf{u}^k$. Therefore, by treating $\mathbf{x}^{k+1}$ as a function of $(\mathbf{z}^k, \mathbf{u}^k)$, we can rewrite $\mathbf{z}^{k+1}$, and subsequently $\mathbf{u}^{k+1}$, as a function of $(\mathbf{z}^k, \mathbf{u}^k)$ as well. In this way, the **x-z-u** iteration can be treated as a fixed-point iteration of $(\mathbf{z}, \mathbf{u})$:

$$(\mathbf{z}^{k+1}, \mathbf{u}^{k+1}) = G(\mathbf{z}^k, \mathbf{u}^k).$$

Similarly, we can treat the **z-x-u** scheme as a fixed-point iteration of $(\mathbf{x}, \mathbf{u})$. In addition, to ensure stability for Anderson acceleration, we should define criteria to evaluate the effectiveness of an accelerated iterate, as well as a fall-back strategy when the criteria are not met. Goldstein et al. [2014] pointed out that if the problem is convex, then its *combined residual* is monotonically decreased by ADMM. For the **x-z-u** iteration, the combined residual is defined as

$$r_{\mathbf{x}\text{-}\mathbf{z}\text{-}\mathbf{u}}^{k+1} = \mu\|\mathbf{A}\mathbf{x}^{k+1} - \mathbf{B}\mathbf{z}^{k+1} - \mathbf{c}\|^2 + \mu\|\mathbf{B}(\mathbf{z}^{k+1} - \mathbf{z}^k)\|^2. \tag{11}$$

Here the first term is a measure of the primal residual, whereas the second term is related to the dual residual (7) but without the matrix

Algorithm 1: Anderson acceleration for ADMM with **x**-**z**-**u** iteration.

```
Data: $\mathbf{x}^0$, $\mathbf{z}^0$, $\mathbf{u}^0$: initial values of variables;
  $L$: the augmented Lagrangian function;
  $m$: the number of previous iterates used for acceleration;
  AA($\mathcal{G}$, $\mathcal{F}$): Anderson accleration from a sequence $\mathcal{G}$ of fixed-point
  mapping results of previous iterates, and a sequence $\mathcal{F}$ of their
  corresponding fixed-point residuals;
  $I_{\max}$ : the maximum number of iterations;
  $\varepsilon$: convergence threshold for combined residual.
$\mathbf{x}_{\text{default}} = \mathbf{x}^0$; $\mathbf{z}_{\text{default}} = \mathbf{z}^0$; $\mathbf{u}_{\text{default}} = \mathbf{u}^0$;
$r_{\text{prev}} = +\infty$; $j = 0$; reset = TRUE; $k = 0$;
while TRUE do
    // Run one iteration of ADMM
    $\mathbf{x}_\star = \text{argmin}_{\mathbf{x}} L(\mathbf{x}, \mathbf{z}^k, \mathbf{u}^k)$;
    $\mathbf{z}_\star = \text{argmin}_{\mathbf{z}} L(\mathbf{x}_\star, \mathbf{z}, \mathbf{u}^k)$;
    $\mathbf{u}_\star = \mathbf{u}^k + \mathbf{A}\mathbf{x}_\star - \mathbf{B}\mathbf{z}_\star - \mathbf{c}$;
    // Compute the combined residual
    $r = \|\mathbf{A}\mathbf{x}_\star - \mathbf{B}\mathbf{z}_\star - \mathbf{c}\|^2 + \|\mathbf{B}(\mathbf{z}_\star - \mathbf{z}^k)\|^2$;
    if reset == TRUE OR $r < r_{\text{prev}}$ then
        // Record the latest accepted iterate
        $\mathbf{x}_{\text{default}} = \mathbf{x}_\star$; $\mathbf{z}_{\text{default}} = \mathbf{z}_\star$; $\mathbf{u}_{\text{default}} = \mathbf{u}_\star$;
        $r_{\text{prev}} = r$; reset = FALSE;
        // Compute the accelerated iterate
        $\mathbf{g}_j = (\mathbf{z}_\star, \mathbf{u}_\star)$;  $\mathbf{f}_j = (\mathbf{z}_\star - \mathbf{z}^k, \mathbf{u}_\star - \mathbf{u}^k)$;
        $j = j + 1$; $\overline{m} = \min(m - 1, j)$;
        $(\mathbf{z}^{k+1}, \mathbf{u}^{k+1}) = \text{AA}\left([\mathbf{g}_j, \ldots, \mathbf{g}_{j-\overline{m}}], [\mathbf{f}_j, \ldots, \mathbf{f}_{j-\overline{m}}]\right)$;
        $k = k + 1$;
    else
        // Revert to the last accepted iterate
        $\mathbf{z}^k = \mathbf{z}_{\text{default}}$; $\mathbf{u}^k = \mathbf{u}_{\text{default}}$; reset = TRUE;
    end if
    if $k \geq I_{\max}$ OR $r < \varepsilon$ then          // Check termination
        return $\mathbf{x}_{\text{default}}$;        // Return the last accepted x
    end if
end while
```

$\mathbf{A}^T$. The combined residual for the **z**-**x**-**u** iteration is defined as

$$r_{\text{z-x-u}}^{k+1} = \mu\|\mathbf{A}\mathbf{x}^{k+1} - \mathbf{B}\mathbf{z}^{k+1} - \mathbf{c}\|^2 + \mu\|\mathbf{A}(\mathbf{x}^{k+1} - \mathbf{x}^k)\|^2. \tag{12}$$

Although [Goldstein et al. 2014] only proved the monotonic decrease of the combined residual for convex problems, our experiments show that the combined residual is decreased by the majority of iterates from the non-convex ADMM solvers considered in this paper. Indeed, if ADMM converges to a solution, then both the primal residual $\mathbf{A}\mathbf{x}^{k+1} - \mathbf{B}\mathbf{z}^{k+1} - \mathbf{c}$ and the variable changes $\mathbf{z}^{k+1} - \mathbf{z}^k$ and $\mathbf{x}^{k+1} - \mathbf{x}^k$ must converge to zero, so the combined residual must converge to zero as well. Therefore, we evaluate the effectiveness of an accelerated iterate by checking whether it decreases the combined residual compared with the previous iteration, and revert to the un-accelerated ADMM iterate if this is not the case.

Algorithm 1 summarizes our Anderson acceleration approach for the **x**-**z**-**u** iteration. Note that the evaluation of combined residual requires computing the change of **z** in one un-accelerated ADMM iteration. However, given an accelerated iterate $(\mathbf{z}_{\text{AA}}, \mathbf{u}_{\text{AA}})$, it is often difficult to find a pair $(\mathbf{z}_\dagger, \mathbf{u}_\dagger)$ that leads to $(\mathbf{z}_{\text{AA}}, \mathbf{u}_{\text{AA}})$ after one

Algorithm 2: Anderson acceleration for ADMM with **z**-**x**-**u** iteration.

```
$\mathbf{x}_{\text{default}} = \mathbf{x}^0$; $\mathbf{u}_{\text{default}} = \mathbf{u}^0$; $r_{\text{prev}} = +\infty$; $j = 0$; reset = TRUE; $k = 0$;
while TRUE do
    $\mathbf{z}_\star = \text{argmin}_{\mathbf{z}} L(\mathbf{x}^k, \mathbf{z}, \mathbf{u}^k)$;
    $\mathbf{x}_\star = \text{argmin}_{\mathbf{x}} L(\mathbf{x}, \mathbf{z}_\star, \mathbf{u}^k)$;
    $\mathbf{u}_\star = \mathbf{u}^k + \mathbf{A}\mathbf{x}_\star - \mathbf{B}\mathbf{z}_\star - \mathbf{c}$;
    $r = \|\mathbf{A}\mathbf{x}_\star - \mathbf{B}\mathbf{z}_\star - \mathbf{c}\|^2 + \|\mathbf{A}(\mathbf{x}_\star - \mathbf{x}^k)\|^2$;
    if reset == TRUE OR $r < r_{\text{prev}}$ then
        $\mathbf{x}_{\text{default}} = \mathbf{x}_\star$; $\mathbf{u}_{\text{default}} = \mathbf{u}_\star$; $r_{\text{prev}} = r$; reset = FALSE;
        $j = j + 1$; $\overline{m} = \min(m - 1, j)$;
        $\mathbf{g}_j = (\mathbf{x}_\star, \mathbf{u}_\star)$;  $\mathbf{f}_j = (\mathbf{x}_\star - \mathbf{x}^k, \mathbf{u}_\star - \mathbf{u}^k)$;
        $(\mathbf{x}^{k+1}, \mathbf{u}^{k+1}) = \text{AA}\left([\mathbf{g}_j, \ldots, \mathbf{g}_{j-\overline{m}}], [\mathbf{f}_j, \ldots, \mathbf{f}_{j-\overline{m}}]\right)$;
        $k = k + 1$;
    else
        $\mathbf{x}^k = \mathbf{x}_{\text{default}}$; $\mathbf{u}^k = \mathbf{u}_{\text{default}}$; reset = TRUE;
    end if
    if $k \geq I_{\max}$ OR $r < \varepsilon$ then
        return $\mathbf{x}_{\text{default}}$;
    end if
end while
```

ADMM iteration (i.e., $(\mathbf{z}_{\text{AA}}, \mathbf{u}_{\text{AA}}) = G(\mathbf{z}_\dagger, \mathbf{u}_\dagger)$). Therefore, we run one ADMM iteration on $(\mathbf{z}_{\text{AA}}, \mathbf{u}_{\text{AA}})$ instead, and use the resulting values $(\mathbf{z}_\star, \mathbf{u}_\star) = G(\mathbf{z}_{\text{AA}}, \mathbf{u}_{\text{AA}})$ to evaluate the combined residual. If the accelerated iterate is accepted, then the computation of $(\mathbf{z}_\star, \mathbf{u}_\star)$ can be reused in the next step of the algorithm and incurs no overhead. We can derive an acceleration method for the **x**-**z**-**u** iteration in a similar way, by swapping **x** and **z** and adopting Eq. (8) for the computation of combined residual, as summarized in Algorithm 2.

Remark 3.1. If the target function Φ contains an indicator function for a hard constraint on the primal variable updated in the second step of an ADMM iteration (i.e., **z** in the **x**-**z**-**u** iteration, or **x** in the **z**-**x**-**u** iteration), then after each iteration this variable must satisfy the hard constraint. However, as Anderson acceleration computes the accelerated iterate via an affine combination of previous iterates, the accelerated $\mathbf{z}_{\text{AA}}$ or $\mathbf{x}_{\text{AA}}$ may violate the constraint unless its feasible set is an affine space. In other words, the accelerated iterate may not correspond to a valid ADMM iteration, and may cause issues if it is used as a solution. Therefore, to apply Anderson acceleration, we should ensure that Φ contains no indicator function associated with the primal variable updated in the second step of the original ADMM iteration. This does not limit the applicability of our method, because it can always be achieved by introducing auxiliary variables and choosing an appropriate iteration scheme. The simulation in Fig. 4 is an example of changing the iteration scheme to allow acceleration.

3.3 ADMM with a separable target function

The general approach in Section 3.2 does not assume any special structure of the target function. When the target function terms for **x** and **z** are separable, it is possible to improve the efficiency of acceleration further. To this end, we consider the following problem

$$\min_{\mathbf{x},\mathbf{z}} \; f(\mathbf{x}) + g(\mathbf{z}), \quad \text{s.t. } \mathbf{A}\mathbf{x} - \mathbf{B}\mathbf{z} = \mathbf{c}. \tag{13}$$

Moreover, we assume this problem satisfies the following properties:

Assumption 3.1. *Matrix* $\mathbf{B}$ *is invertible.*

Assumption 3.2. $f(\mathbf{x})$ *is a strongly convex quadratic function*

$$f(\mathbf{x}) = \frac{1}{2}(\mathbf{x} - \tilde{\mathbf{x}})^T \mathbf{G}(\mathbf{x} - \tilde{\mathbf{x}}), \tag{14}$$

where $\tilde{\mathbf{x}}$ *is a constant and* $\mathbf{G}$ *is a symmetric positive definite matrix.*

One example of such optimization is the implicit time integration of elastic bodies in [Overby et al. 2017], where $\tilde{\mathbf{x}}$ is the predicted values of node positions $\mathbf{x}$ without internal forces, $\mathbf{G} = \mathbf{M}/\Delta t^2$ where $\mathbf{M}$ is the mass matrix and Δt is the integration time step, the auxiliary variable $\mathbf{z}$ stacks the deformation gradient of each element, and $g(\mathbf{z})$ sums the elastic potential energy for all elements. For the problem (13), the $\mathbf{x}$-$\mathbf{z}$-$\mathbf{u}$ iteration of ADMM becomes

$$\mathbf{x}^{k+1} = (\mathbf{G} + \mu\mathbf{A}^T\mathbf{A})^{-1}(\mathbf{G}\tilde{\mathbf{x}} + \mu\mathbf{A}^T(\mathbf{B}\mathbf{z}^k + \mathbf{c} - \mathbf{u}^k)), \tag{15}$$

$$\mathbf{z}^{k+1} = \operatorname*{argmin}_{\mathbf{z}} \left(g(\mathbf{z}) + \frac{\mu}{2}\|\mathbf{A}\mathbf{x}^{k+1} - \mathbf{B}\mathbf{z} - \mathbf{c} + \mathbf{u}^k\|^2 \right), \tag{16}$$

$$\mathbf{u}^{k+1} = \mathbf{u}^k + \mathbf{A}\mathbf{x}^{k+1} - \mathbf{B}\mathbf{z}^{k+1} - \mathbf{c}. \tag{17}$$

And the $\mathbf{z}$-$\mathbf{x}$-$\mathbf{u}$ iteration becomes

$$\mathbf{z}^{k+1} = \operatorname*{argmin}_{\mathbf{z}} \left(g(\mathbf{z}) + \frac{\mu}{2}\|\mathbf{A}\mathbf{x}^{k} - \mathbf{B}\mathbf{z} - \mathbf{c} + \mathbf{u}^k\|^2 \right), \tag{18}$$

$$\mathbf{x}^{k+1} = (\mathbf{G} + \mu\mathbf{A}^T\mathbf{A})^{-1}(\mathbf{G}\tilde{\mathbf{x}} + \mu\mathbf{A}^T(\mathbf{B}\mathbf{z}^{k+1} + \mathbf{c} - \mathbf{u}^k)), \tag{19}$$

$$\mathbf{u}^{k+1} = \mathbf{u}^k + \mathbf{A}\mathbf{x}^{k+1} - \mathbf{B}\mathbf{z}^{k+1} - \mathbf{c}. \tag{20}$$

Similar to Remark 3.1, we assume that the target function contains no indicator function for the primal variable updated in the second step. The general acceleration algorithms in Section 3.2 treat ADMM as a fixed-point iteration of $(\mathbf{z}, \mathbf{u})$ or $(\mathbf{x}, \mathbf{u})$. Next, we will show that if the problem satisfies certain conditions, then ADMM becomes a fixed-point iteration of only one variable, allowing us to reduce the overhead of Anderson acceleration and improve its effectiveness.

Remark 3.2. Without assuming the convexity of function $g(\cdot)$, there may be multiple solutions for the minimization problems in (16) and (18). Throughout this paper, we assume the solver adopts a deterministic algorithm for (16) and (18), so that given the same values of $\mathbf{x}$ and $\mathbf{u}$ it always returns the same value of $\mathbf{z}$.

3.3.1 $\mathbf{x}$-$\mathbf{z}$-$\mathbf{u}$ *iteration.* For the $\mathbf{x}$-$\mathbf{z}$-$\mathbf{u}$ iteration (15)-(17), under certain conditions $\mathbf{u}^{k+1}$ can be represented as a function of $\mathbf{z}^{k+1}$:

Proposition 3.1. *If the optimization problem* (13) *satisfies Assumptions 3.1 and 3.2, and the function* $g(\mathbf{z})$ *is differentiable, then the* $\mathbf{x}$-$\mathbf{z}$-$\mathbf{u}$ *iteration* (15)-(17) *satisfies*

$$\mathbf{u}^{k+1} = \frac{1}{\mu}\mathbf{B}^{-T}\nabla g(\mathbf{z}^{k+1}). \tag{21}$$

A proof is given in Appendix A. Proposition 3.1 shows that $\mathbf{u}^{k+1}$ can be recovered from $\mathbf{z}^{k+1}$. Therefore, we can treat the $\mathbf{x}$-$\mathbf{z}$-$\mathbf{u}$ iteration (15)-(17) as a fixed-point iteration of $\mathbf{z}$ instead of $(\mathbf{z}, \mathbf{u})$, and apply Anderson acceleration to $\mathbf{z}$ alone. From the accelerated $\mathbf{z}_{\mathrm{AA}}$, we recover its corresponding dual variable $\mathbf{u}_{\mathrm{AA}}$ via Eq. (21). This approach brings two major benefits. First, the main computational overhead for Anderson acceleration in each iteration is to update the normal equation system for the problem (10), which involves inner products of time complexity $O(mn)$ where n is the dimension of variables that undergo fixed-point iteration [Peng et al. 2018]. Since $\mathbf{B}$ is invertible, $\mathbf{u}$ and $\mathbf{z}$ are of the same dimension; thus this new approach reduces the computational cost of inner products by half. Another benefit is a more simple criterion for the effectiveness of an accelerated iterate, based on the following property:

Algorithm 3: Anderson acceleration for ADMM with $\mathbf{x}$-$\mathbf{z}$-$\mathbf{u}$ iteration, on a problem (13) that satisfies Assumptions 3.1, 3.2 and with a differentiable g.

```
1  $r_{\mathrm{prev}} = +\infty$;  $j = 0$;  reset = TRUE;  $k = 0$;
2  while TRUE do
       // Update x with (15) and compute residual with (22)
3      $\mathbf{x}^{k+1} = (\mathbf{G} + \mu\mathbf{A}^T\mathbf{A})^{-1}(\mathbf{G}\tilde{\mathbf{x}} + \mu\mathbf{A}^T(\mathbf{B}\mathbf{z}^k + \mathbf{c} - \mathbf{u}^k))$;
4      $r = \|\mathbf{A}\mathbf{x}^{k+1} - \mathbf{B}\mathbf{z}^k - \mathbf{c}\|$;
5      if reset == FALSE AND $r \geq r_{\mathrm{prev}}$ then     // Check residual
6          $\mathbf{z}^k = \mathbf{z}_{\mathrm{default}}$;          // Revert to un-accelerated z
           // Re-compute u and x with (17) and (15)
7          $\mathbf{u}^k = \mathbf{u}^{k-1} + \mathbf{A}\mathbf{x}^k - \mathbf{B}\mathbf{z}^k - \mathbf{c}$;
8          $\mathbf{x}^{k+1} = (\mathbf{G} + \mu\mathbf{A}^T\mathbf{A})^{-1}(\mathbf{G}\tilde{\mathbf{x}} + \mu\mathbf{A}^T(\mathbf{B}\mathbf{z}^k + \mathbf{c} - \mathbf{u}^k))$;
           // Re-compute residual
9          $r = \|\mathbf{A}\mathbf{x}^{k+1} - \mathbf{B}\mathbf{z}^k - \mathbf{c}\|$;  reset = TRUE;
10     end if
       // Check termination criteria
11     if $k + 1 \geq I_{\max}$ OR $r < \varepsilon$ then
12         return $\mathbf{x}^{k+1}$;
13     end if
       // Compute un-accelerated z value with (16)
14     $\mathbf{z}_{\mathrm{default}} = \mathrm{argmin}_{\mathbf{z}} \left( g(\mathbf{z}) + \frac{\mu}{2}\|\mathbf{A}\mathbf{x}^{k+1} - \mathbf{B}\mathbf{z} - \mathbf{c} + \mathbf{u}^k\|^2 \right)$
       // Compute accelerated z value
15     $j = j + 1$;  $\overline{m} = \min(m, j)$;
16     $\mathbf{g}_j = \mathbf{z}_{\mathrm{default}}$;   $\mathbf{f}_j = \mathbf{z}_{\mathrm{default}} - \mathbf{z}^k$;
17     $\mathbf{z}^{k+1} = \mathrm{AA}([\mathbf{g}_j, \ldots, \mathbf{g}_{j-\overline{m}}], [\mathbf{f}_j, \ldots, \mathbf{f}_{j-\overline{m}}])$;
       // Recover compatible u value with (21)
18     $\mathbf{u}^{k+1} = \frac{1}{\mu}\mathbf{B}^{-T}\nabla g(\mathbf{z}^{k+1})$;
19     $k = k + 1$;  $r_{\mathrm{prev}} = r$;
20 end while
```

Proposition 3.2. *Suppose the problem* (13) *satisfies Assumptions 3.1 and 3.2, and the function* $g(\mathbf{z})$ *is differentiable. Let* $\mathbf{z}^{k+1} = G_{xzu}(\mathbf{z}^k)$ *denote the fixed-point iteration of* $\mathbf{z}$ *induced by the* $\mathbf{x}$-$\mathbf{z}$-$\mathbf{u}$ *iteration* (15)-(17). *Then* $\mathbf{z}^{k+1}$ *is a fixed point of mapping* $G_{xzu}(\cdot)$ *if and only if*

$$\mathbf{A}\mathbf{x}^{k+2} - \mathbf{B}\mathbf{z}^{k+1} - \mathbf{c} = \mathbf{0}. \tag{22}$$

A proof is given in Appendix B. Note that the left-hand side of (22) has a similar form as the primal residual, but involves the value of $\mathbf{x}$ in the next iteration. Accordingly, we evaluate the effectiveness of an accelerated iterate $\mathbf{z}_{\mathrm{AA}}$ and its corresponding dual variable $\mathbf{u}_{\mathrm{AA}}$ by first computing a new value $\mathbf{x}_\star$ according to the $\mathbf{x}$-update step (15), then evaluating a residual $\hat{\mathbf{r}}_{\mathbf{x}\text{-}\mathbf{z}\text{-}\mathbf{u}} = \mathbf{A}\mathbf{x}_\star - \mathbf{B}\mathbf{z}_{\mathrm{AA}} - \mathbf{c}$. We only accept $\mathbf{z}_{\mathrm{AA}}$ if it leads to a smaller norm of this residual compared to the previous iteration; otherwise, we revert to the last un-accelerated iterate. If $\mathbf{z}_{\mathrm{AA}}$ is accepted, then $\mathbf{x}_\star$ can be reused in the next step. The main benefit here is that we do not need to run an additional ADMM iteration to verify the effectiveness of $\mathbf{z}_{AA}$, which incurs less computational overhead when the accelerated iterate is rejected. This acceleration strategy is summarized in Algorithm 3. Fig. 2 shows an example where accelerating $\mathbf{z}$ alone leads to a faster decrease of combined residual than accelerating $\mathbf{z}, \mathbf{u}$ together.

3.3.2 **z-x-u** *iteration.* Similar to the previous discussion, when the problem satisfies certain conditions, the **z-x-u** scheme is a fixed-point iteration of only one variable. In particular, we have:

Proposition 3.3. *If the optimization problem* (13) *satisfies Assumptions 3.1 and 3.2, then the* **z-x-u** *iteration* (18)-(20) *satisfies*

$$\mathbf{x}^{k+1} = \tilde{x} - \mu \mathbf{G}^{-1}\mathbf{A}^T\mathbf{u}^{k+1}. \tag{23}$$

A proof is given in Appendix C. This property implies that $\mathbf{x}^{k+1}$ can be recovered from $\mathbf{u}^{k+1}$; thus we can treat the **z-x-u** scheme (18)-(20) as a fixed-point iteration of $\mathbf{u}$ instead of $(\mathbf{x}, \mathbf{u})$. In theory, we can apply Anderson acceleration to the history of $\mathbf{u}$ to obtain an accelerated iterate $\mathbf{u}_{AA}$, and recover the corresponding $\mathbf{x}_{AA}$ from Eq. (23). However, this would require solving a linear system with matrix $\mathbf{G}$, and can be computationally expensive. Instead, we note that $\mathbf{x}^{k+1}$ and $\mathbf{u}^{k+1}$ are related to by an affine map, and this relation is satisfied by any previous pair of $\mathbf{x}$ and $\mathbf{u}$ values. Then since $\mathbf{u}_{AA}$ is an affine combination of previous $\mathbf{u}$ values, we can apply the same affine combination coefficients to the corresponding previous $\mathbf{x}$ values to obtain $\mathbf{x}_{AA}$, which is guaranteed to satisfy Eq. (23) with $\mathbf{u}_{AA}$. As the affine combination coefficients are computed from $\mathbf{u}$ only, this still reduces the computational cost compared to applying Anderson acceleration to $(\mathbf{x}, \mathbf{u})$. Similar to the **x-z-u** case, we can verify the convergence of the **z-x-u** iteration by comparing $\mathbf{x}$ in the current iteration with the value of $\mathbf{z}$ in the next iteration:

Proposition 3.4. *Suppose the problem* (13) *satisfies Assumptions 3.1 and 3.2. Let* $\mathbf{u}^{k+1} = G_{zxu}(\mathbf{u}^k)$ *denote the fixed-point iteration of* $\mathbf{u}$ *induced by the* **x-z-u** *iteration* (18)-(20). *Then* $\mathbf{u}^{k+1}$ *is a fixed point of mapping* $G_{zxu}(\cdot)$ *if and only if*

$$\mathbf{A}\mathbf{x}^{k+1} - \mathbf{B}\mathbf{z}^{k+2} - \mathbf{c} = \mathbf{0}. \tag{24}$$

Accordingly, we evaluate the effectiveness of $\mathbf{u}_{AA}$ and $\mathbf{x}_{AA}$ by computing from them a $\mathbf{z}_\star$ using Eq. (18), and evaluating the residual $\hat{\mathbf{r}}_{\text{z-x-u}} = \mathbf{A}\mathbf{x}_{AA} - \mathbf{B}\mathbf{z}_\star - \mathbf{c}$. We accept $\mathbf{u}_{AA}$ if the norm of this residual is smaller than the previous iteration, and revert to the last un-accelerated iterate otherwise. If $\mathbf{u}_{AA}$ is accepted, then $\mathbf{z}_\star$ is reused in the next step. Algorithm 4 summarizes our approach.

Remark 3.3. We have shown that ADMM can be reduced to a fixed-point iteration of the second primal variable or the dual variable based on Assumptions 3.1 and 3.2, and (for the **x-z-u** iteration) the smoothness of g. In fact, these assumptions can be further relaxed. We refer the reader to Appendix E for more details. Fig. 11 is an example of using such relaxed conditions to reduce the fixed-point iteration to one variable.

Algorithm 4: Anderson acceleration for ADMM with **z-x-u** iteration, on a problem (13) that satisfies Assumptions 3.1 and 3.2.

```
1  $r_{\text{prev}} = +\infty$;  $j = 0$;  reset = TRUE;  $k = 0$;
2  while TRUE do
       // Update z with (18) and compute residual with (24)
3      $\mathbf{z}^{k+1} = \text{argmin}_{\mathbf{z}} \left(g(\mathbf{z}) + \frac{\mu}{2}\|\mathbf{A}\mathbf{x}^k - \mathbf{B}\mathbf{z} - \mathbf{c} + \mathbf{u}^k\|^2\right)$;
4      $r = \|\mathbf{A}\mathbf{x}^k - \mathbf{B}\mathbf{z}^{k+1} - \mathbf{c}\|$;
       // Check whether the residual increases
5      if reset == FALSE AND $r \geq r_{\text{prev}}$ then
           // Revert to un-accelerated x, u
6          $\mathbf{x}^k = \mathbf{x}_{\text{default}}$;  $\mathbf{u}^k = \mathbf{u}_{\text{default}}$;
7          $\mathbf{z}^{k+1} = \text{argmin}_{\mathbf{z}} \left(g(\mathbf{z}) + \frac{\mu}{2}\|\mathbf{A}\mathbf{x}^k - \mathbf{B}\mathbf{z} - \mathbf{c} + \mathbf{u}^k\|^2\right)$;
8          $\text{r} = \|\mathbf{A}\mathbf{x}^k - \mathbf{B}\mathbf{z}^{k+1} - \mathbf{c}\|$;
9          reset = TRUE;
10     end if
11     if $k + 1 \geq I_{\max}$ OR $r < \varepsilon$ then
12         return $\mathbf{x}^k$;
13     end if
       // Compute un-accelerated x and u
14     $\mathbf{x}_{\text{default}} = (\mathbf{G} + \mu\mathbf{A}^T\mathbf{A})^{-1}(\mathbf{G}\tilde{\mathbf{x}} + \mu\mathbf{A}^T(\mathbf{B}\mathbf{z}^{k+1} + \mathbf{c} - \mathbf{u}^k))$;
15     $\mathbf{u}_{\text{default}} = \mathbf{u}^k + \mathbf{A}\mathbf{x}_{\text{default}} - \mathbf{B}\mathbf{z}^{k+1} - \mathbf{c}$;
       // Use history of u to compute affine coeffients
16     $j = j + 1$;  $\overline{m} = \min(m, j)$;
17     $\mathbf{g}^{\mathbf{x}}_j = \mathbf{x}_{\text{default}}$;  $\mathbf{g}^{\mathbf{u}}_j = \mathbf{u}_{\text{default}}$;  $\mathbf{f}^{\mathbf{u}}_j = \mathbf{u}_{\text{default}} - \mathbf{u}^k$;
18     $(\theta_1^*, \ldots, \theta_{\overline{m}}^*) = \underset{(\theta_1, \ldots, \theta_{\overline{m}})}{\text{argmin}} \left\|\mathbf{f}^{\mathbf{u}}_j - \sum_{i=1}^{\overline{m}} \theta_i(\mathbf{f}^{\mathbf{u}}_{j-i+1} - \mathbf{f}^{\mathbf{u}}_{j-i})\right\|^2$;
       // Compute accelerated x and u with the coefficients
19     $\mathbf{x}^{k+1} = \mathbf{g}^{\mathbf{x}}_j - \sum_{i=1}^{\overline{m}} \theta_i^* \left(\mathbf{g}^{\mathbf{x}}_{j-i+1} - \mathbf{g}^{\mathbf{x}}_{j-i}\right)$;
20     $\mathbf{u}^{k+1} = \mathbf{g}^{\mathbf{u}}_j - \sum_{i=1}^{\overline{m}} \theta_i^* \left(\mathbf{g}^{\mathbf{u}}_{j-i+1} - \mathbf{g}^{\mathbf{u}}_{j-i}\right)$;
21     $k = k + 1$;  $r_{\text{prev}} = r$;
22 end while
```

3.4 Convergence analysis

For Anderson acceleration to be applicable, an ADMM solver must be convergent already. However, many ADMM solvers used in computer graphics lack a convergence guarantee due to the non-convexity of the problems they solve. Although ADMM works well for many non-convex problems in practice, convergence proofs on such problems rely on strong assumptions that are often not satisfied by graphics problems [Li and Pong 2015; Hong et al. 2016; Magnússon et al. 2016; Wang et al. 2019]. In this subsection, we discuss the convergence of ADMM on the problem (13) where the term g in the target function can be non-convex. We first provide a set of conditions for linear convergence of ADMM on such problems, and then give more general convergence proofs using weaker assumptions than existing results in the literature. As the problem structure (13) is common in computer graphics, our new results can potentially expand the applicability of ADMM for graphics problems.

To ease the presentation, we first introduce some notation. To account for the fact that the target function may be unbounded from above due to an indicator function, we suppose all the functions are mappings to $\mathbb{R} \bigcup \{+\infty\}$. Following [Rockafellar 1997], for a function

F we define its effective domain and level set as:

$$\mathrm{dom}(F) := \{\mathbf{x} \mid f(\mathbf{x}) < +\infty\},$$
$$\mathscr{L}_\alpha^F := \{\mathbf{x} \mid f(\mathbf{x}) \leq \alpha\}, \text{ given } \alpha \in \mathbb{R}.$$

A function F is *level-bounded* if $\mathscr{L}_\alpha^F$ is a bounded set for any $\alpha \in \mathbb{R}$. Given a set $\mathcal{S}$, let $\mathcal{I}_{\mathcal{S}}$ and $\mathcal{B}_{\mathcal{S}}$ denote the interior and the boundary of $\mathcal{S}$, respectively. A function F is continuous on $\mathbb{R}^n$ if: (i) it is continuous within $\mathcal{I}_{\mathrm{dom}(F)}$ in the conventional sense; and (ii) $\forall \mathbf{x}_k \to \mathbf{x} \in \mathcal{B}_{\mathrm{dom}(F)}$, we have $F(\mathbf{x}_k) \to F(\mathbf{x}) = +\infty$. We say a function is *Lipschitz differentiable* if it is differentiable and its gradient is Lipschitz continuous. Unless specified otherwise, $\mathbf{I}$ denotes the identity matrix and the identity map. The symbol $\mathrm{conv}(\mathcal{S})$ denotes the convex hull of a set $\mathcal{S}$, and ∂F denotes the set of all sub-differentials for a function F (see [Rockafellar and Wets 2009, Definition 8.3(b)]). For matrix $\mathbf{Q}$, we use $\rho(\mathbf{Q})$ to represent its spectral radius. We will discuss conditions for the ADMM iterates $\{(\mathbf{x}^k, \mathbf{z}^k, \mathbf{u}^k)\}$ to converge to a stationary point $(\mathbf{x}^*, \mathbf{z}^*, \mathbf{u}^*)$ of the augmented Lagrangian for problem (13), which is defined by the conditions [Boyd et al. 2011]:

$$\mathbf{A}\mathbf{x}^* - \mathbf{B}\mathbf{z}^* = \mathbf{c}, \quad \mathbf{0} \in \partial f(\mathbf{x}^*) + \mathbf{A}^T\mathbf{u}^*, \quad \mathbf{0} \in \partial g(\mathbf{z}^*) - \mathbf{B}^T\mathbf{u}^*. \tag{25}$$

Linear convergence. Our discussion involves the following definitions related to the problem (13) and Assumptions 3.1 and 3.2:

$$\hat{g}(\mathbf{z}) := g(\mathbf{B}^{-1}\mathbf{z}), \quad \mathbf{K} := \mathbf{A}\mathbf{G}^{-1}\mathbf{A}^T. \tag{26}$$

We denote by $\rho(\mathbf{K})$ the spectral radius of matrix $\mathbf{K}$. To prove linear convergence of ADMM for the problem (13) regardless of its initial value, we need the following assumption:

Assumption 3.3. *$\nabla\hat{g}$ is Lipschitz differentiable on $\mathbb{R}^n$ with a Lipschitz constant L, i.e. $\|\nabla\hat{g}(\mathbf{z}_1) - \nabla\hat{g}(\mathbf{z}_2)\| \leq L\|\mathbf{z}_1 - \mathbf{z}_2\|$ $\forall \mathbf{z}_1, \mathbf{z}_1 \in \mathbb{R}^n$.*

Then we have:

Theorem 3.1. *If Assumptions 3.1-3.3 are satisfied and $\rho(\mathbf{K}) < \frac{1}{2L}$, then for a sufficiently large μ the* **x**-**z**-**u** *iteration* (15)-(17) *converges to a stationary point defined in Eq.* (25). *Moreover,*

$$\|\mathbf{B}\mathbf{z}^{n+1} - \mathbf{B}\mathbf{z}^n\| \leq \gamma_1\|\mathbf{B}\mathbf{z}^n - \mathbf{B}\mathbf{z}^{n-1}\|,$$

where $\gamma_1 = \dfrac{\frac{\mu\rho(\mathbf{K})}{1+\mu\rho(\mathbf{K})} + \frac{L}{\mu}}{1 - \frac{L}{\mu}} < 1$ *is a constant.*

Theorem 3.2. *If Assumptions 3.1-3.3 are satisfied, $\rho(\mathbf{K}) < \frac{1}{L}$ and $\mathbf{I} - \mu\mathbf{K}$ is invertible, then for a sufficiently large μ the* **z**-**x**-**u** *iteration* (18)-(20) *converges to a stationary point defined in Eq.* (25). *Moreover,*

$$\|\mathbf{v}^{k+1} - \mathbf{v}^k\| \leq \gamma_2\|\mathbf{v}^k - \mathbf{v}^{k-1}\|,$$

where $\mathbf{v}^k = (\mathbf{I} - \mu\mathbf{K})\mathbf{u}^k$ *and* $\gamma_2 = \frac{\mu\rho(\mathbf{K})}{1+\mu\rho(\mathbf{K})} + \frac{L}{\mu - L} < 1$.

Proofs are provided in Appendix F. The theorems above rely on Assumption 3.3 which requires the function g to be globally Lipschitz differentiable. This may not be the case for some graphics problems. For example, the StVK energy used for simulation of hyperelastic materials is a quartic function of the deformation gradient, and is locally Lipschitz differentiable but not globally so. For such problems, we can still prove linear convergence with additional conditions on its initial value and penalty parameter. In the following, we use $T(\mathbf{x}, \mathbf{z})$ to denote the target function (13). We make the following relaxed assumption about $\hat{g}$:

Assumption 3.4. *(1) $\hat{g}$ is level-bounded, and $\hat{g}(\mathbf{z}) \geq 0$ $\forall \mathbf{z} \in \mathbb{R}^n$.*
(2) $\hat{g}$ is continuous on $\mathbb{R}^n$ and differentiable in $\mathcal{I}_{\mathrm{dom}(\hat{g})}$.
(3) $\hat{g}$ is Lipschitz differentiable on any compact convex set in $\mathrm{dom}(\hat{g})$.

For linear convergence of the **x**-**z**-**u** iteration, we assume the following for the initial value $(\mathbf{x}^0, \mathbf{z}^0, \mathbf{u}^0)$ and penalty parameter μ:

Assumption 3.5. *(1) $\mathbf{z}^0 = \mathbf{B}^{-1}(\mathbf{A}\mathbf{x}^0 - \mathbf{c}), \mathbf{u}^0 = \frac{1}{\mu}\mathbf{B}^{-T}\nabla g(\mathbf{z}^0)$. $\mathbf{z}^0 \in dom(g)$.*
(2) μ is large enough such that $c_1 \leq 1$, where

$$c_1 = \sup_{\mathbf{z} \in \mathscr{L}^g_{T^0+1}} \frac{1}{2\mu}\|\mathbf{B}^{-T}\nabla g(\mathbf{z})\|^2$$

and $T^0 = T(\mathbf{x}^0, \mathbf{z}^0)$. Moreover, suppose $\mathrm{conv}(\mathscr{L}^{\hat{g}}_{T^0+c_1}) \subset \mathrm{dom}(\hat{g})$ and let L_c be a Lipschitz constant of $\nabla\hat{g}$ over this set.

Theorem 3.3. *Suppose Assumptions 3.1, 3.2, 3.4, 3.5 are satisfied, $\frac{\mu}{2} - \frac{L_c^2}{\mu} > \frac{L_c}{2}$, and $\rho(\mathbf{K}) < \frac{1}{2L_c}$. Then for a sufficiently large μ the* **x**-**z**-**u** *iteration* (15)-(17) *converges to a stationary point defined in Eq.* (25),
and $\|\mathbf{B}\mathbf{z}^{n+1} - \mathbf{B}\mathbf{z}^n\| \leq \gamma_3\|\mathbf{B}\mathbf{z}^n - \mathbf{B}\mathbf{z}^{n-1}\|$ *with* $\gamma_3 = \dfrac{\frac{\mu\rho(\mathbf{K})}{1+\mu\rho(\mathbf{K})} + \frac{L_c}{\mu}}{1 - \frac{L_c}{\mu}} < 1$.

For the **z**-**x**-**u** iteration, we need a different assumption that relies on the following proposition which is proved in Appendix F.3:

Proposition 3.5. *Let $R(\mathbf{A})$ be the range of matrix $\mathbf{A}$. Then for any $\mathbf{x} \in R(\mathbf{A})$, $\|\mathbf{K}\mathbf{x}\| \geq \eta\|\mathbf{x}\|$ where $\eta > 0$ is a constant depending on $\mathbf{K}$.*

Assumption 3.6. *The initial value $(\mathbf{x}^0, \mathbf{z}^0, \mathbf{u}^0)$ satisfies:*
(1) $\mathbf{z}^0 = \mathbf{B}^{-1}(\mathbf{A}\mathbf{x}^0 - \mathbf{c}), \mathbf{x}^0 = \tilde{\mathbf{x}}, \mathbf{u}^0 = 0$. $\mathbf{z}^0 \in dom(g)$.
(2) μ is large enough such that $c_2 + c_3 \leq 1$, where

$$c_2 = \sup_{(\mathbf{x},\mathbf{z}) \in \mathscr{L}^T_{T^0+1}} \frac{2}{\eta^2\mu}\|\mathbf{A}\mathbf{x} - \mathbf{A}\tilde{\mathbf{x}}\|^2 + \left(\frac{2\rho(\mathbf{K})^2}{\mu\eta^2} + \frac{1}{\mu}\right)\|\mathbf{B}^{-T}\nabla g(\mathbf{z})\|^2,$$

$$c_3 = \left(\frac{8\rho(\mathbf{K})^2}{\mu\eta^2} + \frac{4}{\mu}\right)\|\mathbf{B}^{-T}\nabla g(\mathbf{z}^0)\|^2,$$

where η is defined in Proposition 3.5. Moreover, let L_d be a Lipschitz constant of $\nabla\hat{g}$ over $\mathrm{conv}(\mathscr{L}^{\hat{g}}_{T^0+c_2+c_3})$, and suppose $\mathrm{conv}(\mathscr{L}^{\hat{g}}_{T^0+c_2+c_3}) \subset \mathrm{dom}(\hat{g})$.

Theorem 3.4. *Suppose Assumptions 3.1, 3.2, 3.4, 3.6 are satisfied, $\rho(\mathbf{K}) < \frac{1}{L_d}$, and $\mathbf{I} - \mu\mathbf{K}$ is invertible. Then for a sufficiently large μ the* **z**-**x**-**u** *iteration* (18)-(20) *converges to a stationary point defined in Eq.* (25), *and* $\|\mathbf{v}^{k+1} - \mathbf{v}^k\| \leq \gamma_4\|\mathbf{v}^k - \mathbf{v}^{k-1}\|$, *with* $\mathbf{v}^k = (\mathbf{I} - \mu\mathbf{K})\mathbf{u}^k$ *and* $\gamma_4 = \frac{\mu\rho(\mathbf{K})}{1+\mu\rho(\mathbf{K})} + \frac{L_d}{\mu - L_d} < 1$.

The proofs for these two theorems are given in Appendix F.

Remark 3.4. Unlike existing linear convergence proofs such as [Lin et al. 2015; Deng and Yin 2016; Giselsson and Boyd 2017], we do not require both f and g to be convex. This makes our proofs applicable to some graphics problems with a non-convex g, such as the elastic body simulation problem in [Overby et al. 2017] where g is an elastic potential energy. In the supplementary material we provide numerical verification of linear convergence on such a problem.

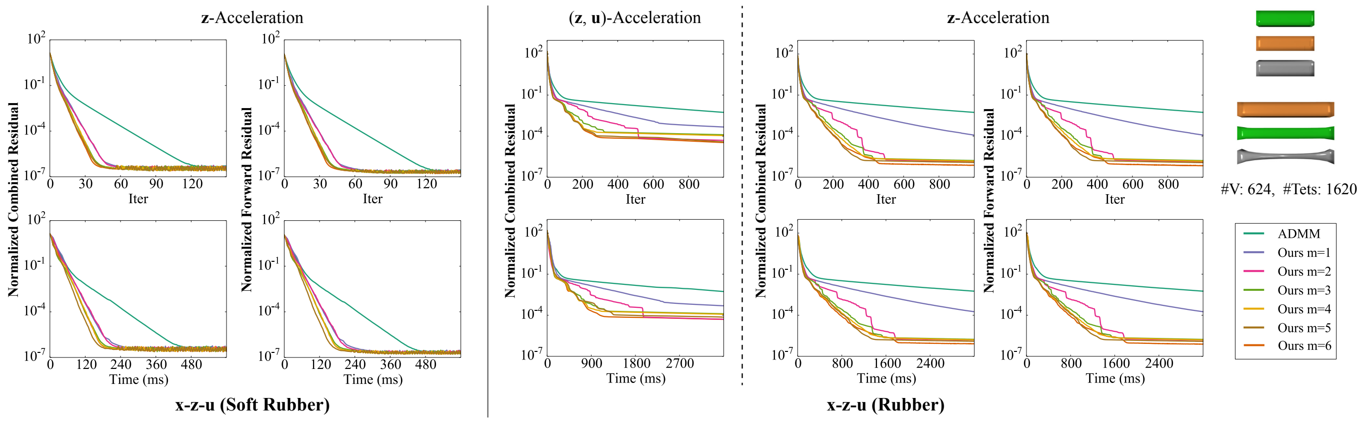

Fig. 2. Comparison between the ADMM solver in [Overby et al. 2017] and our method according to Algorithm 3, for computing the same frame of a simulation sequence with three elastic bars. Two material stiffness settings ("soft rubber" and "rubber") are used for testing. In both case, our method leads to faster decrease of residuals and accelerates the convergence. For the case with rubber, we also test Algorithm 1 that applies Anderson acceleration to (**z**, **u**), which also speeds up the convergence but is less effective than accelerating **z** alone.

General convergence under weak assumptions. If a linear convergence rate is not needed, the assumptions above can be further relaxed to prove the convergence of ADMM on problem (13): instead of the relation between the matrix **K** and the Lipschitz constant L, we require the following weak assumption on function g.

ASSUMPTION 3.7. *g is a semi-algebraic function.*

A function $F : \mathbb{R}^n \mapsto \mathbb{R}$ is called semi-algebraic if its graph $\{(\mathbf{y}, F(\mathbf{y})) \mid \mathbf{y} \in \mathbb{R}^n\} \subset \mathbb{R}^{n+1}$ is a union of finitely many sets each defined by a finite number of polynomial equalities and strict inequalities [Li and Pong 2015]. This assumption covers a large range of functions used in computer graphics. For example, polynomials (such as StVK energy) and rational functions (such as NURBS) are both semi-algebraic. Then we have:

THEOREM 3.5. *Suppose Assumptions 3.1, 3.2, 3.4, 3.5 and 3.7 are satisfied, and $\frac{\mu}{2} - \frac{L_c^2}{\mu} > \frac{L_c}{2}$. Then for a sufficiently large μ the* **x**-**z**-**u** *iteration* (15)-(17) *converges to a stationary point defined in Eq.* (25), *and $\sum_{n=1}^{+\infty} \|\mathbf{z}^{k+1} - \mathbf{z}^k\| < \infty$.*

THEOREM 3.6. *If Assumptions 3.1, 3.2, 3.4, 3.6 and 3.7 are satisfied, then for a sufficiently large μ the* **z**-**x**-**u** *iteration* (18)-(20) *converges to a stationary point defined in Eq.* (25), *and $\sum_{n=1}^{+\infty} \|\mathbf{A}\mathbf{x}^{k+1} - \mathbf{A}\mathbf{x}^k\| < \infty$.*

Proofs are given in Appendix F.1 and F.3.

Remark 3.5. Compared with existing convergence results for nonconvex ADMM such as [Li and Pong 2015; Wang et al. 2019], for the **x**-**z**-**u** iteration we do not require the function g to be globally Lipschitz differentiable, and for the **z**-**x**-**u** iteration we do not require the matrix **A** to be of full row rank. This makes our results applicable to a wider range of problems in computer graphics. In particular, for geometry optimization, the reduction matrix **A** that relates vertex positions to auxiliary variables may not be of full row rank, potentially due to the presence of auxiliary variables that are derived in the same way from vertex positions but involved in different constraints. Although for the **z**-**x**-**u** iteration our assumptions on g are more restrictive than those in [Li and Pong 2015; Wang et al. 2019], such assumptions are still general enough to be satisfied by many graphics problems.

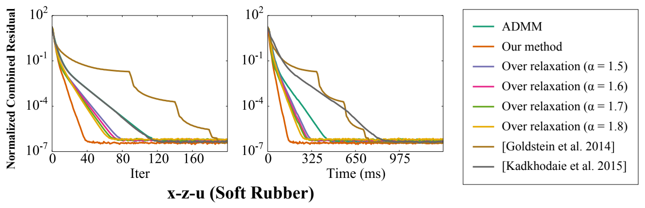

Fig. 3. Comparison with other ADMM acceleration schemes on the same non-convex problem for rubber simulation as Fig. 2. The methods from [Goldstein et al. 2014] and [Kadkhodaie et al. 2015], which are designed for convex problems, are ineffective for this problem instance. Over-relaxation is effective in accelerating the convergence, but not as much as our approach.

4 RESULTS

We apply our methods to a variety of ADMM solvers in graphics. We implement Anderson acceleration following the source code released by the authors of [Peng et al. 2018][1]. The source code of our implementation is available at https://github.com/bldeng/AA-ADMM. All examples are run on a desktop PC with 32GB of RAM and a quad-core CPU at 3.6GHz. To account for the dimension and the numerical range of the variables, we use the following *normalized combined residual* R_c and *normalized forward residual* R_f to measure convergence:

$$R_c = \sqrt{\frac{r_c}{N_{\mathbf{z}} \cdot a^2}}, \quad R_f = \sqrt{\frac{r_f}{N_{\mathbf{z}} \cdot a^2}}, \tag{27}$$

where r_c is the combined residual computed from Eq. (11) or (12), r_f is the squared norm of the residual of Eq. (22) or (24), $N_{\mathbf{z}}$ is the dimension of **z**, and $a > 0$ is a scalar that indicates the typical

[1]https://github.com/bldeng/AASolver

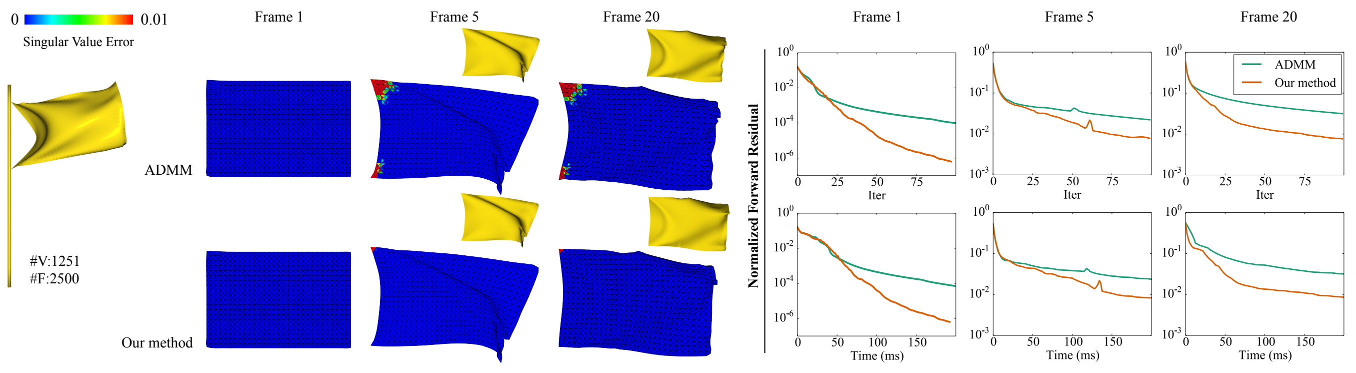

Fig. 4. For the simulation of a discretized flag with hard constraints that limit its strain, our accelerated solver convergences faster than an ADMM solver. Here the color-coding shows the deviation from the deformation gradient singular values from their prescribed range. Using the same computational budget to compute a frame, the results with our solver satisfy the strain limiting constraints better.

variable range. In the following, for all physical simulation and geometry optimization problems, we set a to the average edge length of the initial discretized model. For image processing problems, we simply set $a = 1$. For the choice of parameter m, similar to [Peng et al. 2018] we observe that a large m tends to improve the reduction of iteration count but increases the computational overhead per iteration (see Fig. 2). We choose $m = 6$ by default.

4.1 Physical simulation

Overby et al. [2017] performed physical simulation via the following optimization problem:

$$\min_{\mathbf{x},\mathbf{z}} \; f(\mathbf{x}) + g(\mathbf{z}) \quad \text{s.t. } \mathbf{W}(\mathbf{z} - \mathbf{D}\mathbf{x}) = 0, \tag{28}$$

Here $\mathbf{x}$ is the node positions of the discretized object, $f(\mathbf{x})$ is a momentum energy of the form (14) with $\mathbf{G}$ being a scaled mass matrix, $\mathbf{D}\mathbf{x}$ collects the deformation gradient of each element, $g(\mathbf{z})$ is the elastic potential energy, and $\mathbf{W}$ is a diagonal scaling matrix that improves conditioning. This problem is solved in (28) using ADMM with the $\mathbf{x}$-$\mathbf{z}$-$\mathbf{u}$ iteration. As it satisfies the assumptions in Proposition 3.1, we apply Anderson acceleration to variable $\mathbf{z}$ according to Algorithm 3. Our method is implemented based on the source code released by the authors of [Overby et al. 2017][2]. Fig. 2 compares the simulation performance on three elastic bars subject to horizontal external forces on their two ends. We use the same material stiffness for all bars, and a different elastic potential energy model for each bar (corotational, StVK and neo-Hookean, respectively). We apply the original solver and our solver with different m values to the same problem for a particular frame, and plot their normalized combined residuals and normalized forward residuals through the iterations. The methods are compared on two types of material stiffness ("soft rubber" and "rubber" as defined in the code from [Overby et al. 2017], with the latter one being stiffer). Our method decreases both residuals much faster than the original ADMM solver for each stiffness settings. Moreover, these two residuals are highly correlated, which demonstrates the effectiveness of using the forward residual to verify accelerated iterates according to Proposition 3.2. On the rubber models, we also evaluate the performance of the general approach in Algorithm 1 that accelerates $\mathbf{z}$ and $\mathbf{u}$ together. We can see that accelerating $\mathbf{z}$ alone leads to a faster decrease of the combined residual. One possible reason is that Algorithm 3 explicitly enforces the compatibility condition (21), so that the accelerated $\mathbf{z}$ and the recovered $\mathbf{u}$ always correspond to a valid intermediate value for a certain ADMM iterate sequence. This property does not hold for the general approach, since it only performs affine combination to obtain the accelerated $\mathbf{z}$ and $\mathbf{u}$, which is more akin to finding a new initial value for an ADMM sequence. In Fig. 3, we use the same soft rubber simulation problem to compare our method with existing ADMM acceleration techniques, including [Goldstein et al. 2014] and [Kadkhodaie et al. 2015] which combined Nesterov's acceleration scheme with a restarting rule based on combined residual, as well as over-relaxation [Eckstein and Bertsekas 1992] with a relaxation parameter $\alpha \in [1.5, 1.8]$ as explained in [Boyd et al. 2011, §3.4.3]. As [Goldstein et al. 2014; Kadkhodaie et al. 2015] rely on the convexity of the problem, they are ineffective for this non-convex problem and in fact increases the computational time. Although over-relaxation speeds up the decrease of residual, it achieves less acceleration than our method.

The solver in [Overby et al. 2017] allows enforcing hard constraints on node positions. Our method can be applied in such cases as well. In Fig. 4, we simulate the movement of a triangulated flag under the wind force. Within $g(\mathbf{z})$, the elastic potential energy for each triangle is defined as the squared Euclidean distance from its deformation gradient to the closest rotation matrix. In addition, $g(\mathbf{z})$ contains an indicator function term for the strain limit of each triangle that requires all singular values of the deformation gradient to be within the range $[0.95, 1.05]$. Due to such hard constraints for $\mathbf{z}$, we cannot apply our method to the $\mathbf{x}$-$\mathbf{z}$-$\mathbf{u}$ iteration (see Remark 3.1). Instead, we adopt the $\mathbf{z}$-$\mathbf{x}$-$\mathbf{u}$ iteration and apply Algorithm 4 to accelerate $\mathbf{u}$ alone, because the iteration satisfies the assumptions in Proposition 3.3. We compare the original ADMM solver with our accelerated solver with $m = 6$. To this end, we first apply our solver to compute a simulation sequence, and then re-solve the optimization problem using the original ADMM solver. Fig. 4 plots the normalized forward residual from each solver on three frames, where we

[2]https://github.com/mattoverby/admm-elastic

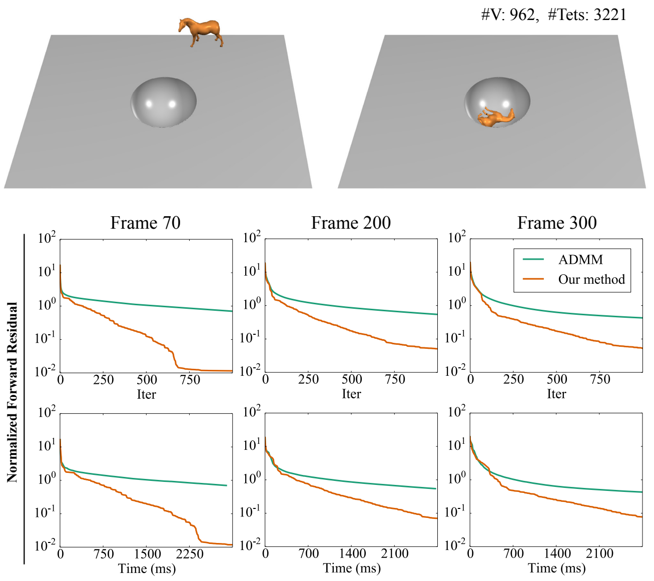

Fig. 5. Simulation of a falling horse, with hard constraints on node positions that prevent them from penetrating the static objects. Our method achieves faster convergence than ADMM, as shown by the plots of normalized forward residual for three frames.

see a faster decrease of the residual using our solver. In addition, for these three frames we take the results from both solvers within the same computational time, and use color-coding to illustrate the maximum deviation of its deformation gradient singular values from the prescribed range on each triangle. We can see that our solver leads to better satisfaction of the strain limiting constraints.

Hard constraints are also used in [Overby et al. 2017] to handle collision between objects. In Figs. 5 and 6, we apply our method in such scenarios. Here an elastic solid horse model falls under gravity and collides with static objects in the scene. In [Overby et al. 2017], this is handled by enforcing hard constraints on $\mathbf{x}$ that prevent the nodes from penetrating the static objects. As this would reduce the $\mathbf{x}$-update step to a time-consuming quadratic programming problem, [Overby et al. 2017] linearizes the constraints and solve the resulting linear system. However, with such modification it is no longer an ADMM algorithm. Therefore, we apply the constraints on $\mathbf{z}$ instead and solve the problem using $\mathbf{z}$-$\mathbf{x}$-$\mathbf{u}$ iteration, with acceleration according to Algorithm 4. Figs. 5 and 6 plot the normalized forward residual for computing certain frames in the simulation sequence, showing a faster decrease of the residual with our method.

4.2 Geometry processing

We also apply our method to an ADMM solver for mesh optimization subject to both soft and hard constraints based on [Deng et al. 2015]. The input is a mesh with vertex positions $\mathbf{x}$, soft constraints $\mathbf{A}_i\mathbf{x} \in C_i$ ($i \in \mathcal{S}$), and hard constraints $\mathbf{A}_j\mathbf{x} \in C_j$ ($j \in \mathcal{H}$). Here each reduction matrix $\mathbf{A}_i$ and $\mathbf{A}_j$ selects vertex positions relevant to the constraint and (where appropriate) compute their differential coordinates with respect to either their mean position or one of the vertices. [Deng et al. 2015] introduce auxiliary variables $\mathbf{z}_i \in C_i$

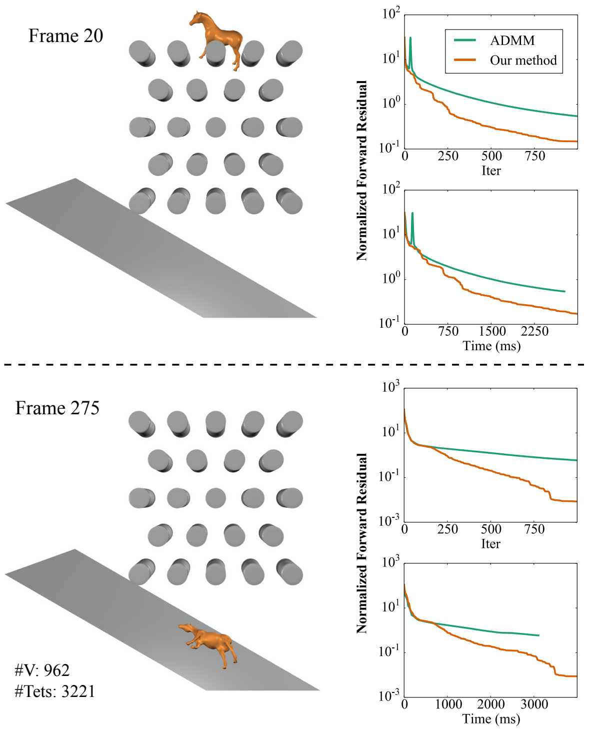

Fig. 6. The same simulation of a falling horse as in Fig. 5, with more complex arrangement of static objects. Our acceleration approach remains effective.

($i \in \mathcal{S}$) and $\mathbf{z}_j \in C_j$ ($j \in \mathcal{H}$) to derive an optimization problem

$$\min_{\mathbf{x},\mathbf{z}} \frac{1}{2}\|\mathbf{L}(\mathbf{x}-\tilde{\mathbf{x}})\|^2 + \sum_{i\in\mathcal{S}}\left(\frac{w_i}{2}\|\mathbf{A}_i\mathbf{x}-\mathbf{z}_i\|^2 + \sigma_{C_i}(\mathbf{z}_i)\right) + \sum_{j\in\mathcal{H}}\sigma_{C_j}(\mathbf{z}_j)$$
$$\text{s.t. } \mathbf{A}_j\mathbf{x}-\mathbf{z}_j = \mathbf{0}, \quad \forall\, j \in \mathcal{H}. \tag{29}$$

Here $\|\mathbf{L}(\mathbf{x}-\tilde{\mathbf{x}})\|^2$ is an optional Laplacian fairing energy for the vertex positions and/or for their displacement from initial positions, whereas $\|\mathbf{A}_i\mathbf{x}-\mathbf{z}_i\|^2$ penalizes the violation of a soft constraint with a user-specified weight w_i. This problem is solved in [Deng et al. 2015] using the augmented Lagrangian method (ALM), where each iteration performs multiple alternate updates of $\mathbf{z}$ and $\mathbf{x}$ followed by a single update of $\mathbf{u}$, using the same formulas as (6). Wu et al. [2011] pointed out that it is more efficient to perform only one alternate update of primal variables per iteration, in which case ALM reduces to ADMM. Therefore, we solve the problem using ADMM with the $\mathbf{z}$-$\mathbf{x}$-$\mathbf{u}$ iteration, and apply the general approach in Algorithm 2 for acceleration because the target function is not separable.

In Fig. 7, we apply our method with $m = 6$ to the wire mesh optimization problem from [Garg et al. 2014]. The input is a regular quad mesh subject to the following constraints:

- Hard constraints: all edges have the same length l; within a face, each angle formed by two incident edges is in the range $[\frac{\pi}{4}, \frac{3\pi}{4}]$.
- Soft constraint: each vertex lies on a given reference surface.

The mesh is optimized without the Laplacian fairing term, i.e., $\mathbf{L} = \mathbf{0}$. Our method leads to a faster decrease of the combined residual with

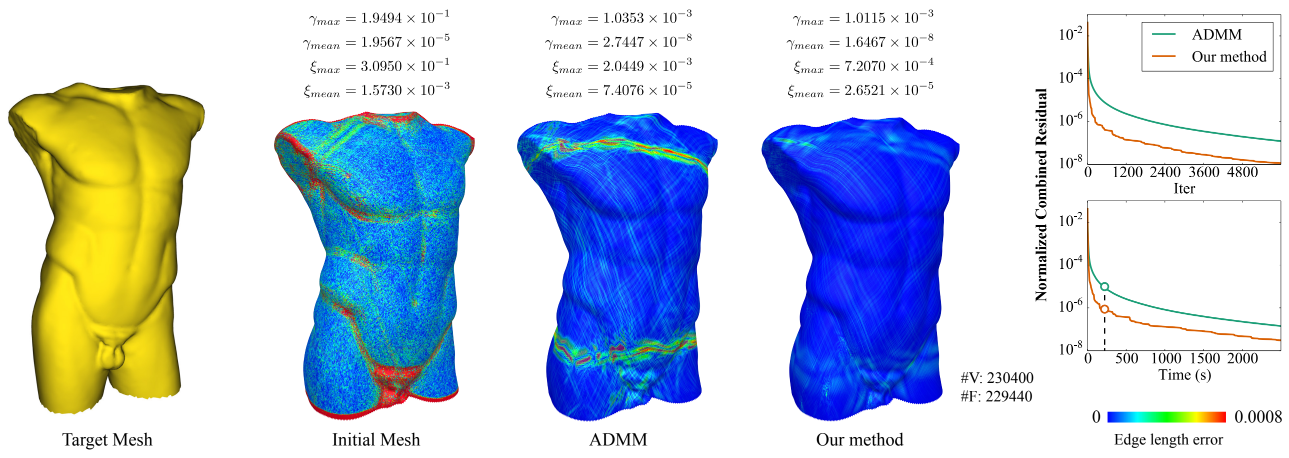
$\gamma_{max} = 1.9494 \times 10^{-1}$
$\gamma_{mean} = 1.9567 \times 10^{-5}$
$\xi_{max} = 3.0950 \times 10^{-1}$
$\xi_{mean} = 1.5730 \times 10^{-3}$
$\gamma_{max} = 1.0353 \times 10^{-3}$
$\gamma_{mean} = 2.7447 \times 10^{-8}$
$\xi_{max} = 2.0449 \times 10^{-3}$
$\xi_{mean} = 7.4076 \times 10^{-5}$
$\gamma_{max} = 1.0115 \times 10^{-3}$
$\gamma_{mean} = 1.6467 \times 10^{-8}$
$\xi_{max} = 7.2070 \times 10^{-4}$
$\xi_{mean} = 2.6521 \times 10^{-5}$
#V: 230400
#F: 229440
Target Mesh
Initial Mesh
ADMM
Our method
ADMM
Our method
Normalized Combined Residual
10^{-2}
10^{-4}
10^{-6}
10^{-8}
0 1200 2400 3600 4800
Iter
0 500 1000 1500 2000
Time (s)
0
0.0008
Edge length error

Fig. 7. Our method accelerates an ADMM solver for wire mesh optimization, as shown by the normalized combined residual plots. We also show two results computed using ADMM and our accelerated solver within the same computational time (indicated in the bottom-right plot), and evaluate their violation of the angle constraints and edge length constraints using the error metrics in Eq. (30). Our result satisfies these constraints better.

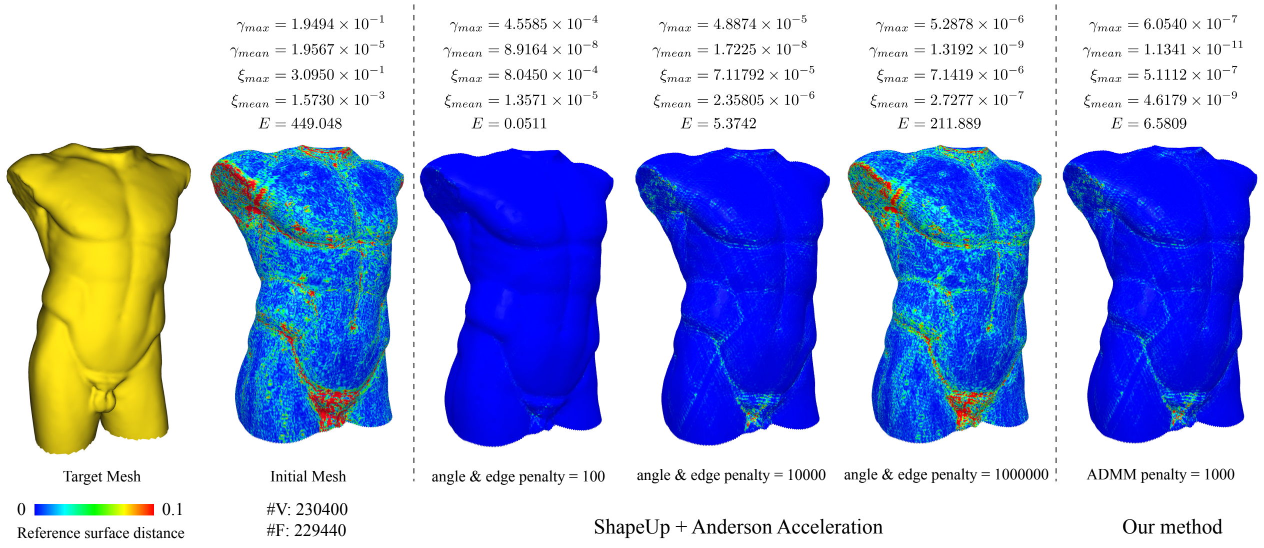
$\gamma_{max} = 1.9494 \times 10^{-1}$
$\gamma_{mean} = 1.9567 \times 10^{-5}$
$\xi_{max} = 3.0950 \times 10^{-1}$
$\xi_{mean} = 1.5730 \times 10^{-3}$
$E = 449.048$
$\gamma_{max} = 4.5585 \times 10^{-4}$
$\gamma_{mean} = 8.9164 \times 10^{-8}$
$\xi_{max} = 8.0450 \times 10^{-4}$
$\xi_{mean} = 1.3571 \times 10^{-5}$
$E = 0.0511$
$\gamma_{max} = 4.8874 \times 10^{-5}$
$\gamma_{mean} = 1.7225 \times 10^{-8}$
$\xi_{max} = 7.11792 \times 10^{-5}$
$\xi_{mean} = 2.35805 \times 10^{-6}$
$E = 5.3742$
$\gamma_{max} = 5.2878 \times 10^{-6}$
$\gamma_{mean} = 1.3192 \times 10^{-9}$
$\xi_{max} = 7.1419 \times 10^{-6}$
$\xi_{mean} = 2.7277 \times 10^{-7}$
$E = 211.889$
$\gamma_{max} = 6.0540 \times 10^{-7}$
$\gamma_{mean} = 1.1341 \times 10^{-11}$
$\xi_{max} = 5.1112 \times 10^{-7}$
$\xi_{mean} = 4.6179 \times 10^{-9}$
$E = 6.5809$
Target Mesh
Initial Mesh
angle & edge penalty = 100
angle & edge penalty = 10000
angle & edge penalty = 1000000
ADMM penalty = 1000
0
0.1
Reference surface distance
#V: 230400
#F: 229440
ShapeUp + Anderson Acceleration
Our method

Fig. 8. Comparison of wire mesh optimzation results using our accelerated ADMM solver and an accelerated quadratic penalty method as described in [Peng et al. 2018]. The error metric E is the sum of squared distances from the mesh vertices to the reference shape, and the color-coding illustrates the distance for each vertex. Although the quadratic penalty method can improve satisfaction of the angle and edge length constraints with a larger penalty weight, this leads to greater deviation from the reference shape.

respect to both the iteration count and the computational time. We also evaluate the violation of hard constraints using the following error metrics for angle α and edge length e:

$$\xi(e) = \frac{|e - l|}{l}, \quad \gamma(\alpha) = \begin{cases} \frac{\pi}{4} - \alpha & \text{if } \alpha < \frac{\pi}{4}, \\ \alpha - \frac{3\pi}{4} & \text{if } \alpha > \frac{3\pi}{4}, \\ 0 & \text{otherwise.} \end{cases} \qquad (30)$$

The data and color-coding in Fig. 7 show that within the same computational time, the result from our method satisfies the hard constraints better than the original ADMM.

Besides ADMM, another popular approach for enforcing hard constraints is the quadratic penalty method, which replaces the original constrained problem by an unconstrained problem with quadratic terms in the target function to penalize the violation of hard constraints [Nocedal and Wright 2006]. Fig. 8 compares the effectiveness of these two approaches in enforcing hard constraints while decreasing the original target function. For the quadratic penalty method, we use ShapeUp [Bouaziz et al. 2012] with Anderson acceleration as described in [Peng et al. 2018], and solve three problem instances with different penalty weights for hard constraints and fixed weights for the other terms. Each solver is run to

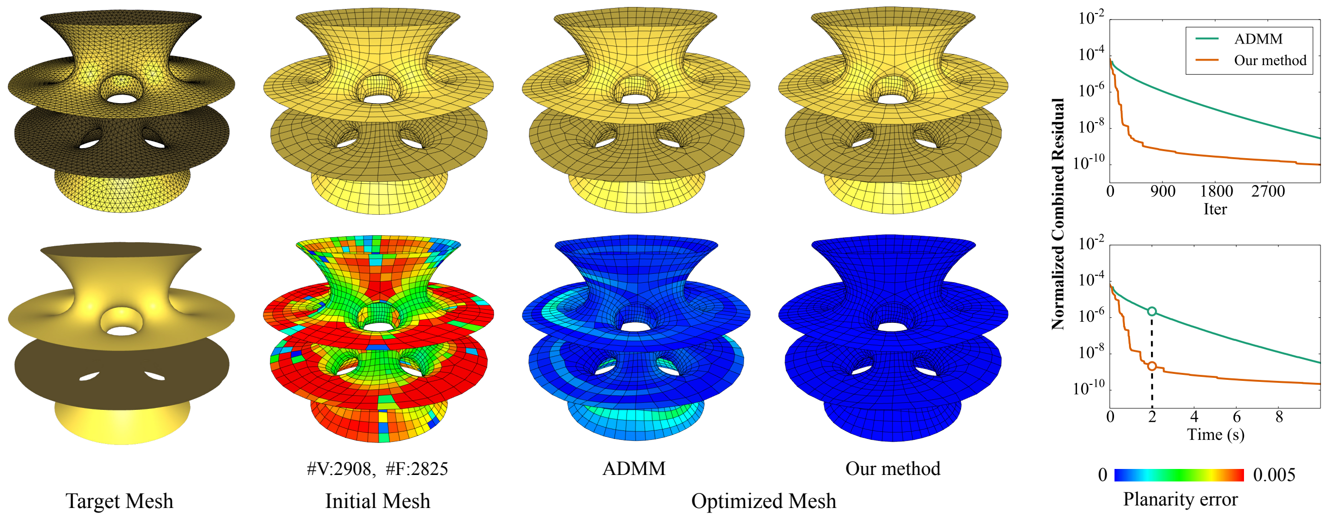

Fig. 9. PQ mesh optimization using our accelerated solver convergences faster than ADMM, and achieves better satisfaction of the planarity constraints within the same computational time (highlighted in the plot in bottom right).

full convergence for comparison. We can see that although a larger penalty weight for hard constraints improves their satisfaction, it also leads to relatively less penalty and greater violation of the soft constraints. In particular, with a large penalty weight to satisfy the hard constraints to a similar level as ADMM, the result from the quadratic penalty method deviates much more from the reference surface than ADMM. It shows that ADMM is more effective in satisfying hard constraints without compromising the minimization of the target function, and our method further improves its efficiency.

In Figs. 1 and 9, we apply our method to planar quad mesh optimization, a classical problem in architectural geometry [Liu et al. 2006]. The input is a quad mesh subject to the following constraints:

- Hard constraint: vertices within each face lie on a common plane.
- Soft constraint: each vertex lies on a given reference surface.

Following [Bouaziz et al. 2012], the reduction matrix for each hard constraint represents the mean-centering operator for the vertices on a common face. The target function includes a Laplacian fairness energy and a relative fairness energy for the vertex positions, as described in [Liu et al. 2011]. We measure the planarity error for each face F of a given mesh using the metric $d_{\max}(F)/\overline{e}$, where $d_{\max}(F)$ is the maximum distance from a vertex of F to the best fitting plane of its vertices, and $\overline{e}$ is the average edge length of the mesh. In both Fig. 1 and Fig. 9, our method accelerates the decrease of the combined residual, producing a result with lower planarity error than the original ADMM within the same computational time.

4.3 Image processing

In Fig. 10, we apply our method to the ADMM solver from the ProxImaL image optimization framework [Heide et al. 2016]. We compare our method with the original solver on the following problem that computes a deconvoluted image $\mathbf{x}$ from an observation image $\mathbf{f}$ with Gaussian noise and a convolution operator $\mathbf{K}$:

$$\min_{\mathbf{x},\mathbf{z}} \lambda_1\|\mathbf{z}_1-\mathbf{f}\|^2+\lambda_2\|\mathbf{z}_2^{i,j}\| \quad \text{s.t.}\, \mathbf{K}\mathbf{x}=\mathbf{z}_1,\ (\nabla\mathbf{x})_{i,j}=\mathbf{z}_2^{i,j}\ \forall i,j, \quad (31)$$

where $(\nabla\mathbf{x})_{i,j}$ is the image gradient of $\mathbf{x}$ at pixel (i,j). This is solved in [Heide et al. 2016] using ADMM with the $\mathbf{x}$-$\mathbf{z}$-$\mathbf{u}$ iteration, and we accelerate it using Algorithm 1 with $m=6$. We modify the source code of the ProxImaL library [3] to implement our accelerated solver, and use conjugate gradient to solve the linear systems in the update steps. Fig. 10 shows that our method requires less computational time and lower iteration count to achieve the same residual value.

Finally, in Fig. 11, we accelerate the ADMM solver used by the Coded Wavefront Sensor in [Wang et al. 2018] for computing the observed wavefront from a captured image. The wavefront $\mathbf{x}$ is computed by solving an optimization problem

$$\min_{\mathbf{x},\mathbf{z}} \lambda\|\nabla\mathbf{x}\|^2+g(\mathbf{z}) \quad \text{s.t.}\ \nabla\mathbf{x}=\mathbf{z}, \quad (32)$$

where $\mathbf{z}$ is an auxiliary variable for image gradient, and $g(\mathbf{z})$ is a quadratic term that measures the consistency between the wavefront and the captured image. From the general condition presented in Appendix E.1, we know that in each iteration the dual variable $\mathbf{u}^k$ can be represented as a function of $\mathbf{z}^k$ via Eq. (49). Therefore, we apply Anderson acceleration to $\mathbf{z}$ alone. Moreover, as $g(\mathbf{z})$ is quadratic, Eq. (49) implies that $\mathbf{z}^k$ and $\mathbf{u}^k$ are related by a linear map. Thus we use the history $\mathbf{z}$ to compute the affine combination coefficients for Anderson acceleration, and apply them to both $\mathbf{z}$ and $\mathbf{u}$ to derive the accelerated $\mathbf{z}$ and its compatible $\mathbf{u}$, similar to Algorithm 4. We modify the source code released by the authors of [Wang et al. 2018] [4] to implement our accelerated solver. Fig. 11 compares the normalized combined residual plots between the two solvers, using a test example provided in the released source code. Compared to the original ADMM, our method leads to a significant reduction of computational time and iteration count for the same accuracy. Also included in the comparison is the GMRES acceleration for ADMM

[3]https://github.com/comp-imaging/ProxImaL
[4]https://github.com/vccimaging/MegapixelAO

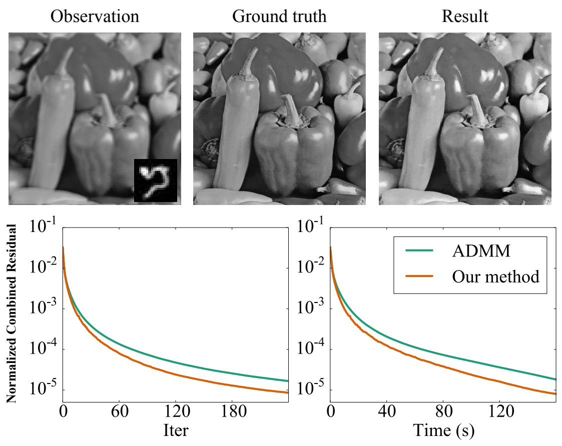

Fig. 10. Our method accelerates the ADMM solver in [Heide et al. 2016] for the deconvolution of a 512×512 image with Gaussian noise using Eq. (31). The given convolution operator $\mathbf{K}$ is visualized in the bottom right of the observation image.

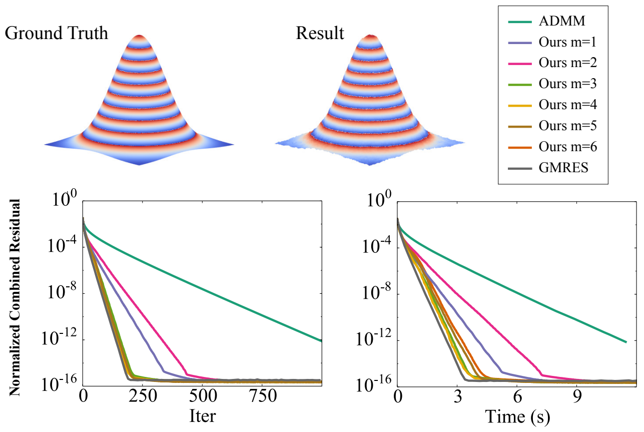

Fig. 11. Our method accelerates the ADMM solver in [Wang et al. 2018] for computing the observed wavefront from a captured image, and achieves similar performance as the specialized GMRES acceleration [Zhang and White 2018] despite being a general acceleration technique.

proposed in [Zhang and White 2018], which is designed specifically for strongly convex quadratic problems. Following [Zhang and White 2018], we restart GMRES every 10 iterations to reduce computational cost. As a general method, our approach is outperformed by GMRES acceleration, but only by a small margin.

5 CONCLUSION AND FUTURE WORK

In this paper, we apply Anderson acceleration to improve the convergence of ADMM on computer graphics problems. We show that ADMM can be interpreted as a fixed-point iteration of the second primal variable and the dual variable in the general case, and of only one of them when the problem has a separable target function and satisfies certain conditions. Such interpretation allows us to directly apply Anderson acceleration in the former case, and further reduce its computational overhead in the latter case. Moreover, for each case we propose a simple residual for measuring the convergence, and use it to determine whether to accept an accelerated iterate. We apply this method to a variety of ADMM solvers in graphics, with applications ranging from physics simulation, geometry processing, to image processing. Our method shows its effectiveness on all these problems, with a notable reduction of iteration account and computational time required to reach the same accuracy. On the theoretical front, we also prove the convergence of ADMM for a common non-convex problem structure in computer graphics under weak assumptions. Our work addresses two main limitations of ADMM especially on non-convex problems, which will help to expand its applicability in computer graphics as a versatile solver for optimization problems that are potentially non-smooth, non-convex, and with hard constraints.

One limitation of our method is that it can be less effective for ADMM solvers with very low computational cost per iteration. In this case, the overhead of Anderson acceleration can cause a large relative increase of computational time, which partly cancels out the speedup gained from the reduction of iteration count. One such example is Fig. 12, where we apply our method to the ADMM solver in [Tao et al. 2019] for correcting a vector field into an integrable gradient field of geodesic distance. Although our method reduces the number of iterations, its large relative overhead actually increases the computational time for achieving the same residual.

Our experiments show that Anderson acceleration is effective in reducing the number of iterations, but we do not have a theoretical guarantee for such property. This is still an open research problem, and the only existing result we are aware of is [Evans et al. 2018], which proves that Anderson acceleration improves the convergence rate for linearly converging fixed-point methods if a set of strong assumptions is satisfied. Further theoretical analysis of our method is needed to understand and guarantee its performance.

Currently we follow the convention and set the mixing parameter $\beta = 1$ for Anderson acceleration. Although it is effective in our experiments, other values of $\beta = 1$ can potentially improve the performance [Eyert 1996]. The optimal choice of mixing parameter remains an open research problem, and should be explored further.

The convergence of ADMM can also be affected by the choice of the penalty parameter and the conditioning of linear side constraints. Recently, researchers have started to analyze the optimal choice of penalty parameter and conditioning for ADMM, but only on simple convex problems [Ghadimi et al. 2015; Giselsson and Boyd 2017]. Overby et al. [2017] proposed a heuristic for choosing such parameters for non-convex physical simulation problems, but there is still no theoretical guarantee for its effectiveness. A potential future research is to perform such analysis on non-convex problems, as well as how they can be used in conjunction with Anderson acceleration to further improve convergence of ADMM.

Finally, as ADMM is a popular solver across different problem domains, we can apply our method to problems outside computer graphics. In this paper we have focused on a problem structure common for graphics tasks. Applications in other domains may involve other problem structures and require different analyses and strategies, which will be an interesting future work.

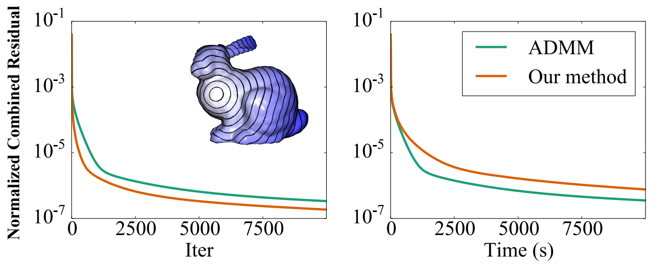

Fig. 12. We apply our method to the ADMM solver in [Tao et al. 2019] for correcting a vector field into an integrable gradient field. Due to the very low computational cost per iteration in the original solver, Anderson acceleration incurs a large relative overhead. As a result, although our method reduces the number of iterations, it actually increases the computational time.

ACKNOWLEDGMENTS

The target model in Figure 7, "Male Torso, Diadumenus Type" by Cosmo Wenman, is licensed under CC BY 3.0. This work was supported by National Natural Science Foundation of China (No. 61672481), and Youth Innovation Promotion Association CAS (No. 2018495).

REFERENCES

M. S. C. Almeida and M. Figueiredo. 2013. Deconvolving images With unknown boundaries using the alternating direction method of multipliers. *IEEE Transactions on Image Processing* 22, 8 (2013), 3074–3086.

Donald G. Anderson. 1965. Iterative procedures for nonlinear integral equations. *J. ACM* 12, 4 (1965), 547–560.

Sofien Bouaziz, Mario Deuss, Yuliy Schwartzburg, Thibaut Weise, and Mark Pauly. 2012. Shape-Up: Shaping discrete geometry with projections. *Comput. Graph. Forum* 31, 5 (2012), 1657–1667.

Sofien Bouaziz, Sebastian Martin, Tiantian Liu, Ladislav Kavan, and Mark Pauly. 2014. Projective dynamics: fusing constraint projections for fast simulation. *ACM Trans. Graph.* 33, 4 (2014), 154:1–154:11.

Sofien Bouaziz, Andrea Tagliasacchi, and Mark Pauly. 2013. Sparse iterative closest point. *Computer Graphics Forum* 32, 5 (2013), 113–123.

Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. 2011. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine learning* 3, 1 (2011), 1–122.

Christopher Brandt, Elmar Eisemann, and Klaus Hildebrandt. 2018. Hyper-reduced projective dynamics. *ACM Trans. Graph.* 37, 4 (2018), 80:1–80:13.

S. H. Chan, X. Wang, and O. A. Elgendy. 2017. Plug-and-play ADMM for image restoration: Fixed-point convergence and applications. *IEEE Transactions on Computational Imaging* 3, 1 (2017), 84–98.

R. Chartrand. 2012. Nonconvex splitting for regularized low-rank + sparse decomposition. *IEEE Transactions on Signal Processing* 60, 11 (2012), 5810–5819.

R. Chartrand and B. Wohlberg. 2013. A nonconvex ADMM algorithm for group sparsity with sparse groups *(ICASSP 2013)*. 6009–6013.

S. Claici, M. Bessmeltsev, S. Schaefer, and J. Solomon. 2017. Isometry-Aware preconditioning for mesh parameterization. *Comput. Graph. Forum* 36, 5 (2017), 37–47.

Patrick L. Combettes and Jean-Christophe Pesquet. 2011. Proximal splitting methods in signal processing. In *Fixed-Point Algorithms for Inverse Problems in Science and Engineering*, Heinz H. Bauschke, Regina S. Burachik, Patrick L. Combettes, Veit Elser, D. Russell Luke, and Henry Wolkowicz (Eds.). 185–212.

Bailin Deng, Sofien Bouaziz, Mario Deuss, Alexandre Kaspar, Yuliy Schwartzburg, and Mark Pauly. 2015. Interactive design exploration for constrained meshes. *Computer-Aided Design* 61, Supplement C (2015), 13–23.

Wei Deng and Wotao Yin. 2016. On the global and linear convergence of the generalized alternating direction method of multipliers. *Journal of Scientific Computing* 66, 3 (2016), 889–916.

Jonathan Eckstein and Dimitri P Bertsekas. 1992. On the Douglas—Rachford splitting method and the proximal point algorithm for maximal monotone operators. *Mathematical Programming* 55, 1-3 (1992), 293–318.

T. Erseghe, D. Zennaro, E. Dall'Anese, and L. Vangelista. 2011. Fast consensus by the alternating direction multipliers method. *IEEE Transactions on Signal Processing* 59, 11 (2011), 5523–5537.

Claire Evans, Sara Pollock, Leo G. Rebholz, and Mengying Xiao. 2018. A proof that Anderson acceleration improves the convergence rate in linearly converging fixed point methods (but not in those converging quadratically). *arXiv preprint arXiv:1810.08455* (2018).

V. Eyert. 1996. A comparative study on methods for convergence acceleration of iterative vector sequences. *J. Comput. Phys.* 124, 2 (1996), 271–285.

Haw-ren Fang and Yousef Saad. 2009. Two classes of multisecant methods for nonlinear acceleration. *Numerical Linear Algebra with Applications* 16, 3 (2009), 197–221.

M. A. T. Figueiredo and J. M. Bioucas-Dias. 2010. Restoration of Poissonian images using alternating direction optimization. *IEEE Transactions on Image Processing* 19, 12 (2010), 3133–3145.

Michel Fortin and Roland Glowinski. 1983. Chapter III On Decomposition-Coordination Methods Using an Augmented Lagrangian. In *Studies in Mathematics and Its Applications*. Vol. 15. Elsevier, 97–146.

Daniel Gabay and Bertrand Mercier. 1976. A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Computers & Mathematics with Applications* 2, 1 (1976), 17–40.

Akash Garg, Andrew O. Sageman-Furnas, Bailin Deng, Yonghao Yue, Eitan Grinspun, Mark Pauly, and Max Wardetzky. 2014. Wire mesh design. *ACM Trans. Graph.* 33, 4 (2014), 66:1–66:12.

E. Ghadimi, A. Teixeira, I. Shames, and M. Johansson. 2015. Optimal parameter selection for the alternating direction method of multipliers (ADMM): quadratic problems. *IEEE Trans. Automat. Control* 60, 3 (2015), 644–658.

P. Giselsson and S. Boyd. 2017. Linear convergence and metric selection for Douglas-Rachford splitting and ADMM. *IEEE Trans. Automat. Control* 62, 2 (2017), 532–544.

T. Goldstein, B. O'Donoghue, S. Setzer, and R. Baraniuk. 2014. Fast alternating direction optimization methods. *SIAM Journal on Imaging Sciences* 7, 3 (2014), 1588–1623.

James Gregson, Ivo Ihrke, Nils Thuerey, and Wolfgang Heidrich. 2014. From capture to simulation: Connecting forward and inverse problems in fluids. *ACM Trans. Graph.* 33, 4 (2014), 139:1–139:11.

D. Hajinezhad, T. Chang, X. Wang, Q. Shi, and M. Hong. 2016. Nonnegative matrix factorization using ADMM: Algorithm and convergence analysis *(ICASSP 2016)*. 4742–4746.

Felix Heide, Steven Diamond, Matthias Nießner, Jonathan Ragan-Kelley, Wolfgang Heidrich, and Gordon Wetzstein. 2016. ProxImaL: Efficient image optimization using proximal algorithms. *ACM Trans. Graph.* 35, 4 (2016), 84:1–84:15.

Nguyenho Ho, Sarah D. Olson, and Homer F. Walker. 2017. Accelerating the Uzawa algorithm. *SIAM Journal on Scientific Computing* 39, 5 (2017), S461–S476.

Mingyi Hong, Zhi-Quan Luo, and Meisam Razaviyayn. 2016. Convergence analysis of alternating direction method of multipliers for a family of nonconvex problems. *SIAM Journal on Optimization* 26, 1 (2016), 337–364.

Y. Hu, D. Zhang, J. Ye, X. Li, and X. He. 2013. Fast and accurate matrix completion via truncated nuclear norm regularization. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35, 9 (2013), 2117–2130.

Mojtaba Kadkhodaie, Konstantina Christakopoulou, Maziar Sanjabi, and Arindam Banerjee. 2015. Accelerated alternating direction method of multipliers *(KDD '15)*. 497–506.

Shahar Z. Kovalsky, Meirav Galun, and Yaron Lipman. 2016. Accelerated quadratic proxy for geometric optimization. *ACM Trans. Graph.* 35, 4 (2016), 134:1–134:11.

Rongjie Lai and Stanley Osher. 2014. A splitting method for orthogonality constrained problems. *Journal of Scientific Computing* 58, 2 (2014), 431–449.

Kenneth Lange. 2004. The MM Algorithm. In *Optimization*. Springer, 119–136.

Guoyin Li and Ting Kei Pong. 2015. Global convergence of splitting methods for nonconvex composite optimization. *SIAM Journal on Optimization* 25, 4 (2015), 2434–2460.

Athanasios P Liavas and Nicholas D Sidiropoulos. 2015. Parallel algorithms for constrained tensor factorization via alternating direction method of multipliers. *IEEE Transactions on Signal Processing* 63, 20 (2015), 5450–5463.

F. Lin, M. Fardad, and M. R. Jovanović. 2013. Design of optimal sparse feedback gains via the alternating direction method of multipliers. *IEEE Trans. Automat. Control* 58, 9 (2013), 2426–2431.

T. Lin, S. Ma, and S. Zhang. 2015. On the global linear convergence of the ADMM with multiBlock variables. *SIAM Journal on Optimization* 25, 3 (2015), 1478–1497.

K. Lipnikov, D. Svyatskiy, and Y. Vassilevski. 2013. Anderson acceleration for nonlinear finite volume scheme for advection-diffusion problems. *SIAM Journal on Scientific Computing* 35, 2 (2013), A1120–A1136.

J. Liu, P. Musialski, P. Wonka, and J. Ye. 2013. Tensor completion for estimating missing values in visual data. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35, 1 (2013), 208–220.

Ligang Liu, Lei Zhang, Yin Xu, Craig Gotsman, and Steven J. Gortler. 2008. A local/global approach to mesh parameterization. *Computer Graphics Forum* 27, 5 (2008), 1495–1504.

Tiantian Liu, Adam W. Bargteil, James F. O'Brien, and Ladislav Kavan. 2013. Fast simulation of mass-spring systems. *ACM Trans. Graph.* 32, 6 (2013), 214:1–214:7.

Tiantian Liu, Sofien Bouaziz, and Ladislav Kavan. 2017. Quasi-Newton methods for real-time simulation of hyperelastic materials. *ACM Trans. Graph.* 36, 3 (2017), 23:1–23:16.

Yang Liu, Helmut Pottmann, Johannes Wallner, Yong-Liang Yang, and Wenping Wang. 2006. Geometric modeling with conical meshes and developable surfaces. *ACM Trans. Graph.* 25, 3 (2006), 681–689.

Yang Liu, Weiwei Xu, Jun Wang, Lifeng Zhu, Baining Guo, Falai Chen, and Guoping Wang. 2011. General planar quadrilateral mesh design using conjugate direction field. *ACM Trans. Graph.* 30, 6 (2011), 140:1–140:10.

Sindri Magnússon, Pradeep Chathuranga Weeraddana, Michael G Rabbat, and Carlo Fischione. 2016. On the convergence of alternating direction Lagrangian methods for nonconvex structured optimization problems. *IEEE Transactions on Control of Network Systems* 3, 3 (2016), 296–309.

Sebastian Martin, Bernhard Thomaszewski, Eitan Grinspun, and Markus Gross. 2011. Example-based elastic materials. *ACM Trans. Graph.* 30, 4 (2011), 72:1–72:8.

Ondrej Miksik, Vibhav Vineet, Patrick Pérez, and Phillip Torr. 2014. Distributed non-convex ADMM-based inference in large-scale random fields *(BMVC 2014)*.

Yurii Nesterov. 1983. A method of solving a convex programming problem with convergence rate $O(1/k^2)$. *Soviet Mathematics Doklady* 27 (1983), 372–376.

Yurii Nesterov. 2013. *Introductory Lectures on Convex Optimization: A Basic Course.* Springer Science & Business Media.

T. Neumann, K. Varanasi, C. Theobalt, M. Magnor, and M. Wacker. 2014. Compressed manifold modes for mesh processing. *Computer Graphics Forum* 33, 5 (2014), 35–44.

Thomas Neumann, Kiran Varanasi, Stephan Wenger, Markus Wacker, Marcus Magnor, and Christian Theobalt. 2013. Sparse localized deformation components. *ACM Trans. Graph.* 32, 6 (2013), 179:1–179:10.

Jorge Nocedal and Stephen J. Wright. 2006. *Numerical Optimization* (2nd ed.). Springer-Verlag New York.

M. Overby, G. E. Brown, J. Li, and R. Narain. 2017. ADMM $\supseteq$ projective dynamics: Fast simulation of hyperelastic models with dynamic constraints. *IEEE Transactions on Visualization and Computer Graphics* 23, 10 (2017), 2222–2234.

Zherong Pan and Dinesh Manocha. 2017. Efficient solver for spacetime control of smoke. *ACM Trans. Graph.* 36, 5 (2017).

Yue Peng, Bailin Deng, Juyong Zhang, Fanyu Geng, Wenjie Qin, and Ligang Liu. 2018. Anderson acceleration for geometry optimization and physics simulation. *ACM Trans. Graph.* 37, 4 (2018), 42:1–42:14.

Florian A. Potra and Hans Engler. 2013. A characterization of the behavior of the Anderson acceleration on linear problems. *Linear Algebra Appl.* 438, 3 (2013), 1002–1011.

Phanisri P. Pratapa, Phanish Suryanarayana, and John E. Pask. 2016. Anderson acceleration of the Jacobi iterative method: An efficient alternative to Krylov methods for large, sparse linear systems. *J. Comput. Phys.* 306 (2016), 43–54.

Péter Pulay. 1980. Convergence acceleration of iterative sequences. the case of SCF iteration. *Chemical Physics Letters* 73, 2 (1980), 393–398.

P. Pulay. 1982. Improved SCF convergence acceleration. *Journal of Computational Chemistry* 3, 4 (1982), 556–560.

Michael Rabinovich, Roi Poranne, Daniele Panozzo, and Olga Sorkine-Hornung. 2017. Scalable locally injective mappings. *ACM Trans. Graph.* 36, 2 (2017), 16:1–16:16.

Ralph Tyrell Rockafellar. 1997. *Convex Analysis.* Princeton University Press.

R Tyrrell Rockafellar and Roger J-B Wets. 2009. *Variational Analysis.* Vol. 317. Springer Science & Business Media.

Thorsten Rohwedder and Reinhold Schneider. 2011. An analysis for the DIIS acceleration method used in quantum chemistry calculations. *Journal of Mathematical Chemistry* 49, 9 (2011), 1889–1914.

Christian Schumacher, Bernhard Thomaszewski, Stelian Coros, Sebastian Martin, Robert Sumner, and Markus Gross. 2012. Efficient simulation of example-based materials *(SCA '12)*. 1–8.

W. Shi, Q. Ling, K. Yuan, G. Wu, and W. Yin. 2014. On the linear convergence of the ADMM in decentralized consensus optimization. *IEEE Transactions on Signal Processing* 62, 7 (2014), 1750–1761.

Anna Shtengel, Roi Poranne, Olga Sorkine-Hornung, Shahar Z. Kovalsky, and Yaron Lipman. 2017. Geometric optimization via composite majorization. *ACM Trans. Graph.* 36, 4 (2017), 38:1–38:11.

A. Simonetto and G. Leus. 2014. Distributed maximum likelihood sensor network localization. *IEEE Transactions on Signal Processing* 62, 6 (2014), 1424–1437.

Olga Sorkine and Marc Alexa. 2007. As-rigid-as-possible surface modeling *(SGP '07)*. 109–116.

H. De Sterck. 2012. A nonlinear GMRES optimization algorithm for canonical tensor decomposition. *SIAM Journal on Scientific Computing* 34, 3 (2012), A1351–A1379.

Phanish Suryanarayana, Phanisri P. Pratapa, and John E. Pask. 2019. Alternating Anderson-Richardson method: An efficient alternative to preconditioned Krylov methods for large, sparse linear systems. *Computer Physics Communications* 234 (2019), 278–285.

J. Tao, J. Zhang, B. Deng, Z. Fang, Y. Peng, and Y. He. 2019. Parallel and scalable heat methods for geodesic distance computation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2019).

Alex Toth, J. Austin Ellis, Tom Evans, Steven Hamilton, C. T. Kelley, Roger Pawlowski, and Stuart Slattery. 2017. Local improvement results for Anderson acceleration with inaccurate function evaluations. *SIAM Journal on Scientific Computing* 39, 5 (2017), S47–S65.

Alex Toth and C. T. Kelley. 2015. Convergence analysis for Anderson acceleration. *SIAM J. Numer. Anal.* 53, 2 (2015), 805–819.

Homer F. Walker and Peng Ni. 2011. Anderson acceleration for fixed-point iterations. *SIAM J. Numer. Anal.* 49, 4 (2011), 1715–1735.

Congli Wang, Qiang Fu, Xiong Dun, and Wolfgang Heidrich. 2018. Megapixel adaptive optics: Towards correcting large-scale distortions in computational cameras. *ACM Trans. Graph.* 37, 4 (2018), 115:1–115:12.

Huamin Wang. 2015. A Chebyshev semi-iterative approach for accelerating projective and position-based dynamics. *ACM Trans. Graph.* 34, 6 (2015), 246:1–246:9.

Huamin Wang and Yin Yang. 2016. Descent methods for elastic body simulation on the GPU. *ACM Trans. Graph.* 35, 6 (2016), 212:1–212:10.

Yu Wang, Wotao Yin, and Jinshan Zeng. 2019. Global convergence of ADMM in nonconvex nonsmooth optimization. *Journal of Scientific Computing* 78, 1 (2019), 29–63.

Zaiwen Wen, Chao Yang, Xin Liu, and Stefano Marchesini. 2012. Alternating direction methods for classical and ptychographic phase retrieval. *Inverse Problems* 28, 11 (2012).

Chunlin Wu, Juyong Zhang, and Xue-Cheng Tai. 2011. Augmented Lagrangian method for total variation restoration with non-quadratic fidelity. *Inverse Problems and Imaging* 5, 1 (2011), 237–261.

Jinhui Xiong, Ramzi Idoughi, Andres A. Aguirre-Pablo, Abdulrahman B. Aljedaani, Xiong Dun, Qiang Fu, Sigurdur T. Thoroddsen, and Wolfgang Heidrich. 2017. Rainbow particle imaging velocimetry for dense 3D fluid velocity imaging. *ACM Trans. Graph.* 36, 4 (2017), 36:1–36:14.

Shiyao Xiong, Juyong Zhang, Jianmin Zheng, Jianfei Cai, and Ligang Liu. 2014. Robust surface reconstruction via dictionary learning. *ACM Trans. Graph.* 33, 6 (2014), 201:1–201:12.

J. Yang, L. Luo, J. Qian, Y. Tai, F. Zhang, and Y. Xu. 2017. Nuclear norm based matrix regression with applications to face recognition with occlusion and illumination changes. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39, 1 (2017), 156–171.

Juyong Zhang, Bailin Deng, Zishun Liu, Giuseppe Patanè, Sofien Bouaziz, Kai Hormann, and Ligang Liu. 2014. Local barycentric coordinates. *ACM Trans. Graph.* 33, 6 (2014), 188:1–188:12.

Junzi Zhang, Brendan O'Donoghue, and Stephen Boyd. 2018. Globally convergent type-I Anderson acceleration for non-smooth fixed-point iterations. *arXiv preprint arXiv:1808.03971* (2018).

Ruiliang Zhang and James T. Kwok. 2014. Asynchronous distributed ADMM for consensus optimization *(ICML '14)*. II–1701–II–1709.

Richard Y. Zhang and Jacob K. White. 2018. GMRES-accelerated ADMM for quadratic objectives. *SIAM Journal on Optimization* 28, 4 (2018), 3025–3056.

Yufeng Zhu, Robert Bridson, and Danny M. Kaufman. 2018. Blended cured quasi-Newton for distortion optimization. *ACM Trans. Graph.* 37, 4 (2018), 40:1–40:14.

A PROOF FOR PROPOSITION 3.1

By the optimality condition of (16) we have:

$$\nabla g(\mathbf{z}^{k+1}) - \mu \mathbf{B}^T(\mathbf{A}\mathbf{x}^{k+1} - \mathbf{B}\mathbf{z}^{k+1} + \mathbf{u}^k - \mathbf{c}) = 0. \quad (33)$$

Put (17) into (33):

$$\nabla g(\mathbf{z}^{k+1}) = \mu \mathbf{B}^T \mathbf{u}^{k+1}, \quad (34)$$

which completes the proof. □

B PROOF FOR PROPOSITION 3.2

For the first part, suppose $\mathbf{z}^{k+1}$ is the fixed-point of the $\mathbf{x}$-$\mathbf{z}$-$\mathbf{u}$ iteration, which means that

$$\mathbf{z}^{k+2} = \mathbf{z}^{k+1}. \quad (35)$$

Then we have

$$\begin{aligned} & \mathbf{u}^{k+2} = \mathbf{u}^{k+1} && \text{by (34)} \\ \Longrightarrow\ & \mathbf{A}\mathbf{x}^{k+2} - \mathbf{B}\mathbf{z}^{k+2} - \mathbf{c} = 0 && \text{by (17)} \\ \Longrightarrow\ & \mathbf{A}\mathbf{x}^{k+2} - \mathbf{B}\mathbf{z}^{k+1} - \mathbf{c} = 0 && \text{by (35).} \end{aligned}$$

For the second part, if $\mathbf{A}\mathbf{x}^{k+2} - \mathbf{B}\mathbf{z}^{k+1} - \mathbf{c} = 0$ then from (17):

$$\mathbf{A}\mathbf{x}^{k+2} - \mathbf{c} + \mathbf{u}^{k+1} = \mathbf{A}\mathbf{x}^{k+1} - \mathbf{c} + \mathbf{u}^k. \quad (36)$$

And from (16) and Remark 3.2

$$\begin{aligned}\mathbf{z}^{k+1} &= \operatorname*{argmin}_{\mathbf{z}} \left(g(\mathbf{z}) + \frac{\mu}{2}\|(\mathbf{A}\mathbf{x}^{k+1} - \mathbf{c} + \mathbf{u}^k) - \mathbf{B}\mathbf{z}\|^2\right)\\ &= \operatorname*{argmin}_{\mathbf{z}} \left(g(\mathbf{z}) + \frac{\mu}{2}\|(\mathbf{A}\mathbf{x}^{k+2} - \mathbf{c} + \mathbf{u}^{k+1}) - \mathbf{B}\mathbf{z}\|^2\right) = \mathbf{z}^{k+2},\end{aligned}$$

which completes the proof. □

C PROOF FOR PROPOSITION 3.3

By (20) we have:

$$\mathbf{B}\mathbf{z}^{k+1} - \mathbf{u}^k + \mathbf{c} = \mathbf{A}\mathbf{x}^{k+1} - \mathbf{u}^{k+1} \tag{37}$$

Put (37) into (19):

$$\begin{aligned}&(\mathbf{G} + \mu\mathbf{A}^T\mathbf{A})x^{k+1} = \mathbf{G}\tilde{\mathbf{x}} + \mu\mathbf{A}^T(\mathbf{A}\mathbf{x}^{k+1} - \mathbf{u}^{k+1})\\ \Longrightarrow\ &\mathbf{G}x^{k+1} = \mathbf{G}\tilde{\mathbf{x}} - \mu\mathbf{A}^T\mathbf{u}^{k+1}\\ \Longrightarrow\ &\mathbf{x}^{k+1} = \tilde{\mathbf{x}} - \mu\mathbf{G}^{-1}\mathbf{A}^T\mathbf{u}^{k+1},\end{aligned} \tag{38}$$

which completes the proof. □

D PROOF FOR PROPOSITION 3.4

For the first part, suppose $\mathbf{u}^{k+1}$ is the fixed-point of the **z**-**x**-**u** iteration, so that

$$\mathbf{u}^{k+2} = \mathbf{u}^{k+1}. \tag{39}$$

Then by (38) and (39):

$$\mathbf{x}^{k+2} = \mathbf{x}^{k+1}. \tag{40}$$

Therefore

$$\begin{aligned}&\mathbf{A}\mathbf{x}^{k+2} - \mathbf{B}\mathbf{z}^{k+2} - \mathbf{c} = \mathbf{0} && \text{by (20) and (39)}\\ \Longrightarrow\ &\mathbf{A}\mathbf{x}^{k+1} - \mathbf{B}\mathbf{z}^{k+2} - \mathbf{c} = 0 && \text{by (40).}\end{aligned}$$

For the second part, suppose

$$\mathbf{A}\mathbf{x}^{k+1} - \mathbf{B}\mathbf{z}^{k+2} - \mathbf{c} = \mathbf{0}. \tag{41}$$

Then we have

$$\begin{aligned}&\mathbf{u}^{k+1} - \mathbf{B}\mathbf{z}^{k+2} = \mathbf{u}^k - \mathbf{B}\mathbf{z}^{k+1} && \text{by (20) and (41)}\\ \Longrightarrow\ &\mathbf{x}^{k+2} = \mathbf{x}^{k+1} && \text{by (19)}\\ \Longrightarrow\ &\mathbf{A}\mathbf{x}^{k+2} - \mathbf{B}\mathbf{z}^{k+2} - \mathbf{c} = \mathbf{0} && \text{by (41)}\\ \Longrightarrow\ &\mathbf{u}^{k+2} = \mathbf{u}^{k+1} && \text{by (20),}\end{aligned}$$

which completes the proof. □

E FURTHER DISCUSSION FOR PROPOSITIONS 3.1-3.4

We now consider the general condition such that between the second updated primal variable and the dual variable, one of them is a function of the other. We consider the most general case:

$$\min_{\mathbf{x},\mathbf{z}}\ f(\mathbf{x}) + g(\mathbf{z}) \quad \text{s.t. } \mathbf{A}\mathbf{x} - \mathbf{B}\mathbf{z} = \mathbf{c}. \tag{42}$$

Unlike Section 3.3, we do not assume any specific form of f and g. We then only need to discuss the following **x**-**z**-**u** iteration because the conclusion for **z**-**x**-**u** iteration is similar:

$$\mathbf{x}^{k+1} \in \operatorname*{argmin}_{\mathbf{x}}\ L(\mathbf{x}, \mathbf{z}^k, \mathbf{u}^k), \tag{43}$$

$$\mathbf{z}^{k+1} \in \operatorname*{argmin}_{\mathbf{z}}\ L(\mathbf{x}^{k+1}, \mathbf{z}, \mathbf{u}^k), \tag{44}$$

$$\mathbf{u}^{k+1} = \mathbf{u}^k + \mathbf{A}\mathbf{x}^{k+1} - \mathbf{B}\mathbf{z}^{k+1} - \mathbf{c}. \tag{45}$$

We first need the subproblem (43) and (44) to be well-defined, for which the next condition is sufficient :

(C1) f and g are bounded from below and lower-semi continuous.

Then we rewrite the ADMM iteration as:

$$-\mathbf{A}^T(\mathbf{A}\mathbf{x}^{k+1} - \mathbf{B}\mathbf{z}^k - \mathbf{c} + \mathbf{u}^k) \in \partial f(\mathbf{x}^{k+1}), \tag{46}$$

$$\mathbf{B}^T(\mathbf{A}\mathbf{x}^{k+1} - \mathbf{B}\mathbf{z}^{k+1} - \mathbf{c} + \mathbf{u}^k) \in \partial g(\mathbf{z}^{k+1}), \tag{47}$$

$$\mathbf{u}^{k+1} = \mathbf{u}^k + \mathbf{A}\mathbf{x}^{k+1} - \mathbf{B}\mathbf{z}^{k+1} - \mathbf{c}. \tag{48}$$

E.1 u as a function of z

By (47) and (48):

$$\mathbf{B}^T\mathbf{u}^{k+1} \in \partial g(\mathbf{z}^{k+1}). \tag{49}$$

Now we can see that $\mathbf{u}^{k+1}$ is a function $\mathbf{z}^{k+1}$ if and only if:

(C2) **B** is invertible.
(C3) $\partial g(\mathbf{z})$ contains exactly one element $\forall \mathbf{z} \in \operatorname{dom}(\partial g)$.

From [Rockafellar and Wets 2009, Theorem 9.18] we know that the next condition is sufficient:

(C3′) $g(\mathbf{z})$ is strictly differentiable $\forall \mathbf{z} \in \operatorname{dom}(\partial g)$.

Moreover, we need additional conditions in order to use Anderson acceleration on **z**. Note that Anderson acceleration generates $\mathbf{z}^{AA}$ by affine combination. So if we want to use (49) to compute $\mathbf{u}_{\mathrm{AA}}$ from $\mathbf{z}_{\mathrm{AA}}$, the following condition is needed:

(C4) The domain of ∂g, defined as $\{\mathbf{z} \mid \partial g(\mathbf{z}) \neq \emptyset\}$, is affine.

E.2 z as a function of u

From (49) we know that **z** is a function of **u** if and only if:

(C5) The inverse mapping of set-valued mapping $\partial g(\mathbf{z})$ is a single-valued mapping.

The next condition is sufficient to ensure (C5) but not necessary:

(C5′) $g(\mathbf{z})$ is strictly convex.

Also, similar to the argument in Appendix E.1, in order to apply Anderson acceleration on **u** we need the following condition:

(C6) The range of ∂g, defined as $\bigcup_{\mathbf{z}\in R^n} \partial g(\mathbf{z})$, is affine.

F PROOFS FOR CONVERGENCE THEOREMS

This section proves the linear convergence theorems when g is locally Lipschitz differentiable (Theorems 3.3 and 3.4) and the general convergence theorems (Theorems 3.5 and 3.6). The proofs for Theorems 3.1 and 3.2 are similar to those for Theorems 3.3 and 3.4, so we will not give their complete proofs but only summarize the main steps. Because of the order in which some lemmas are used in the proofs, we will prove Theorem 3.5 and 3.3 first. Without loss of generality, we assume $\mathbf{c} = \mathbf{0}$ in Eq. (13) to simplify notation.

F.1 Proof for Theorem 3.5

Recall that Theorem 3.5 is about general convergence of the **x**-**z**-**u** iteration. We first note that:

$$\nabla \hat{g}(\mathbf{z}) = \mathbf{B}^{-T}\nabla g(\mathbf{B}^{-1}\mathbf{z}) \tag{50}$$

$$\Longrightarrow \nabla \hat{g}(\mathbf{Bz}) = \mathbf{B}^{-T}\nabla g(\mathbf{z}). \tag{51}$$

These two equations will be used frequently in the following. Note that from Assumption 3.4(2) we can derive (33) from (16). Moreover, based on the definition of L_c in Assumption 3.5 we have:

Proposition F.1. *Suppose the Lipschitz constant of* $\nabla\hat{g}(\mathbf{z})$ *over* $\mathrm{conv}(\mathscr{L}_\alpha^{\hat{g}})$ *is* L_1*, then* $\forall\ \mathbf{Bz}_1, \mathbf{Bz}_2 \in \mathscr{L}_\alpha^{\hat{g}}$*, we have*

$$|\hat{g}(\mathbf{Bz}_1) - \hat{g}(\mathbf{Bz}_2) - \langle\nabla\hat{g}(\mathbf{Bz}_2), \mathbf{Bz}_1 - \mathbf{Bz}_2\rangle| \leq \frac{L_1}{2}\|\mathbf{Bz}_1 - \mathbf{Bz}_2\|^2. \tag{52}$$

Moreover, if $\mu > L_1$*, and* $\mathbf{z}_2 \in \mathrm{argmin}_{\mathbf{z}}(g(\mathbf{z}) + \frac{\mu}{2}\|\mathbf{Bz} - \mathbf{q}\|^2)$*, then:*

$$g(\mathbf{z}_2) + \frac{\mu}{2}\|\mathbf{Bz}_2 - \mathbf{q}\|^2 \leq g(\mathbf{z}_1) + \frac{\mu}{2}\|\mathbf{Bz}_1 - \mathbf{q}\|^2 - \frac{\mu - L_1}{2}\|\mathbf{Bz}_1 - \mathbf{Bz}_2\|^2.$$

The proof is standard so we omit it. Also see [Nesterov 2013, Lemma 1.2.3 & Theorem 2.1.8]. The next lemma is important.

Lemma F.1. *If Assumption 3.4 and 3.5 hold,* $\frac{\mu}{2} - \frac{L_c^2}{\mu} > \frac{L_c}{2}$*, and the* **x**-**z**-**u** *iteration satisfies* $g(\mathbf{z}^k) \leq T(\mathbf{x}^0, \mathbf{z}^0) + c_1$ *and* $L(\mathbf{x}^k, \mathbf{z}^k, \mathbf{u}^k) \leq L(\mathbf{x}^0, \mathbf{z}^0, \mathbf{u}^0) = T(\mathbf{x}^0, \mathbf{z}^0)$*. Then*

$$g(\mathbf{z}^{k+1}) \leq T(\mathbf{x}^0, \mathbf{z}^0) + c_1,\ L(\mathbf{x}^{k+1}, \mathbf{z}^{k+1}, \mathbf{u}^{k+1}) \leq T(\mathbf{x}^0, \mathbf{z}^0). \tag{53}$$

Proof. By the definition of $\mathbf{z}^{k+1}$ in (16):

$$g(\mathbf{z}^{k+1}) + \frac{\mu}{2}\|\mathbf{Ax}^{k+1} - \mathbf{Bz}^{k+1} + \mathbf{u}^k\|^2 \leq g(\mathbf{z}^k) + \frac{\mu}{2}\|\mathbf{Ax}^{k+1} - \mathbf{Bz}^k + \mathbf{u}^k\|^2.$$

And notice the definition of $\mathbf{x}^{k+1}$ in (15):

$$f(\mathbf{x}^{k+1}) + \frac{\mu}{2}\|\mathbf{Ax}^{k+1} - \mathbf{Bz}^k + \mathbf{u}^k\|^2 \leq f(\mathbf{x}^k) + \frac{\mu}{2}\|\mathbf{Ax}^k - \mathbf{Bz}^k + \mathbf{u}^k\|^2. \tag{54}$$

Combine the two equations above:

$$T(\mathbf{x}^{k+1}, \mathbf{z}^{k+1}) + \frac{\mu}{2}\|\mathbf{Ax}^{k+1} - \mathbf{Bz}^{k+1} + \mathbf{u}^k\|^2 \leq L(\mathbf{x}^k, \mathbf{z}^k, \mathbf{u}^k) + \frac{\mu}{2}\|\mathbf{u}^k\|^2.$$

By (17) and (34):

$$T(\mathbf{x}^{k+1}, \mathbf{z}^{k+1}) + \frac{\mu}{2}\|\mathbf{u}^{k+1}\|^2 \leq L(\mathbf{x}^k, \mathbf{z}^k, \mathbf{u}^k) + \frac{1}{2\mu}\|\mathbf{B}^{-T}\nabla g(\mathbf{z}^k)\|^2. \tag{55}$$

Notice that $L(\mathbf{x}^k, \mathbf{z}^k, \mathbf{u}^k) \leq T(\mathbf{x}^0, \mathbf{z}^0)$ and by the definition of c_1:

$$g(\mathbf{z}^{k+1}) \leq T(\mathbf{x}^0, \mathbf{z}^0) + c_1.$$

Thus we have proved the first part. For the second part, we have:

$$L(\mathbf{x}^{k+1}, \mathbf{z}^k, \mathbf{u}^k) \leq L(\mathbf{x}^k, \mathbf{z}^k, \mathbf{u}^k), \tag{56}$$

$$L(\mathbf{x}^{k+1}, \mathbf{z}^{k+1}, \mathbf{u}^k) \leq L(\mathbf{x}^{k+1}, \mathbf{z}^k, \mathbf{u}^k) - \frac{\mu - L_c}{2}\|\mathbf{Bz}^{k+1} - \mathbf{Bz}^k\|^2, \tag{57}$$

$$L(\mathbf{x}^{k+1}, \mathbf{z}^{k+1}, \mathbf{u}^{k+1}) = L(\mathbf{x}^{k+1}, \mathbf{z}^{k+1}, \mathbf{u}^k) + \mu\|\mathbf{u}^{k+1} - \mathbf{u}^k\|^2. \tag{58}$$

Here (56) is derived from (54), (57) is derived from Assumption 3.5(2) and Proposition F.1, and (58) is trivial. Add them together, and then use (34) and the fact that $\frac{\mu}{2} - \frac{L_c^2}{\mu} > \frac{L_c}{2}$:

$$\begin{aligned} L(\mathbf{x}^{k+1}, \mathbf{z}^{k+1}, \mathbf{u}^{k+1}) &\leq L(\mathbf{x}^k, \mathbf{z}^k, \mathbf{u}^k) - (\frac{\mu}{2} - \frac{L_c^2}{\mu} - \frac{L_c}{2})\|\mathbf{Bz}^{k+1} - \mathbf{Bz}^k\|^2 \\ &\leq T(\mathbf{x}^0, \mathbf{z}^0), \end{aligned} \tag{59}$$

Which completes the proof. □

From Assumption 3.5(1) and Lemma F.1, we have:

Proposition F.2. *Suppose Assumptions 3.4 and 3.5 hold, and* $\frac{\mu}{2} - \frac{L_c^2}{\mu} > \frac{L_c}{2}$*. Then the* **x**-**z**-**u** *iteration satisfies*

$$g(\mathbf{z}^k) \leq T(\mathbf{x}^0, \mathbf{z}^0) + c_1,\ L(\mathbf{x}^k, \mathbf{z}^k, \mathbf{u}^k) \leq T(\mathbf{x}^0, \mathbf{z}^0). \tag{60}$$

By Proposition F.2, Assumption 3.4(3) has the same effect as the Lipschitz differentiability assumption. The next step is similar to the convergence proof in [Wang et al. 2019], which requires the following properties for the sequence $(\mathbf{x}^k, \mathbf{z}^k, \mathbf{u}^k)$:

(P1) *Boundedness*: the generated sequence $(\mathbf{x}^k, \mathbf{z}^k, \mathbf{u}^k)$ is bounded, and $L(\mathbf{x}^k, \mathbf{z}^k, \mathbf{u}^k)$ is lower bounded.

(P2) *Sufficient descent*: there is a constant $C_1(\mu) > 0$ such that for sufficiently large k, we have:

$$\begin{aligned} &L(\mathbf{x}^k, \mathbf{z}^k, \mathbf{u}^k) - L(\mathbf{x}^{k+1}, \mathbf{z}^{k+1}, \mathbf{u}^{k+1}) \\ &\geq C_1(\mu)(\|\mathbf{B}(\mathbf{z}^{k+1} - \mathbf{z}^k)\|^2 + \|\mathbf{A}(\mathbf{x}^{k+1} - \mathbf{x}^k)\|^2). \end{aligned}$$

(P3) *Subgradient bound*: there is a constant $C_2(\mu) > 0$ and $\mathbf{d}^{k+1} \in \partial L(\mathbf{x}^{k+1}, \mathbf{y}^{k+1}, \mathbf{u}^{k+1})$ such that

$$\|\mathbf{d}^{k+1}\| \leq C_2(\mu)(\|\mathbf{B}(\mathbf{z}^{k+1} - \mathbf{z}^k)\| + \|\mathbf{A}(\mathbf{x}^{k+1} - \mathbf{x}^k)\|).$$

(P4) *Limiting continuity*: if $(\mathbf{x}^*, \mathbf{z}^*, \mathbf{u}^*)$ is the limit point of the sub-sequence $(\mathbf{x}^{k_s}, \mathbf{z}^{k_s}, \mathbf{u}^{k_s})$ for $s \in \mathbb{N}$, then we have:

$$\lim_{s\to\infty} L(\mathbf{x}^{k_s}, \mathbf{z}^{k_s}, \mathbf{u}^{k_s}) = L(\mathbf{x}^*, \mathbf{z}^*, \mathbf{u}^*).$$

Note that although the **x**-**z**-**u** iteration is not same as the one defined in [Li and Pong 2015], the proof for [Li and Pong 2015, Theorem 3] is not affected by the difference. Combining it with [Wang et al. 2019, Proposition 2], we can prove Theorem 3.5:

Proof for Theorem 3.5. From [Wang et al. 2019, Proposition 2], [Li and Pong 2015, Theorem 3], and Proposition F.2 in our paper, we only need to show (P1)-(P4) hold for $(\mathbf{x}^k, \mathbf{z}^k, \mathbf{u}^k)$.

For (P1), from (55) we have:

$$T(\mathbf{x}^k, \mathbf{z}^k) \leq T(\mathbf{x}^0, \mathbf{z}^0) + c_1. \tag{61}$$

From Assumption 3.4(1) $g(\mathbf{z})$ is level-bounded and **G** is invertible so $f(\mathbf{x})$ is also level-bounded, thus $(\mathbf{x}^k, \mathbf{z}^k)$ is bounded. The boundedness of $\mathbf{u}^k$ can be derived from (33). The lower boundedness of $L(\mathbf{x}^k, \mathbf{z}^k, \mathbf{u}^k)$ comes from Assumption 3.5(2) and the fact that $T(\mathbf{x}, \mathbf{z}) \geq 0$. In fact we have: $L(\mathbf{x}^k, \mathbf{z}^k, \mathbf{u}^k) \geq -c_1$.

In the derivation of (56), we did not use the fact that $f(\mathbf{x})$ is quadratic. If we take this into consideration, then (56) becomes:

$$L(\mathbf{x}^{k+1}, \mathbf{z}^k, \mathbf{u}^k) \leq L(\mathbf{x}^k, \mathbf{z}^k, \mathbf{u}^k) - l\|\mathbf{x}^{k+1} - x^k\|^2. \tag{62}$$

Here $l > 0$ is some constant. (62),(57) and (58) show that (P2) holds.
(P4) is trivial for our problem. For (P3) we have:

$$\begin{aligned}\nabla_{\mathbf{x}} L(\mathbf{x}^{k+1}, \mathbf{z}^{k+1}, \mathbf{u}^{k+1}) &= \mathbf{G}(\mathbf{x}^{k+1} - \tilde{\mathbf{x}}) + \mu \mathbf{A}^T(\mathbf{A}\mathbf{x}^{k+1} - \mathbf{B}\mathbf{z}^{k+1} + \mathbf{u}^{k+1}) \\ &= \mu \mathbf{A}^T(\mathbf{B}\mathbf{z}^k - \mathbf{B}\mathbf{z}^{k+1} + \mathbf{u}^{k+1} - \mathbf{u}^k), \quad (63)\\ \nabla_{\mathbf{z}} L(\mathbf{x}^{k+1}, \mathbf{z}^{k+1}, \mathbf{u}^{k+1}) &= \nabla g(\mathbf{z}^{k+1}) + \mu \mathbf{B}^T(\mathbf{B}\mathbf{z}^{k+1} - \mathbf{A}\mathbf{x}^{k+1} - \mathbf{u}^{k+1}) \\ &= \mu \mathbf{B}^T(\mathbf{u}^k - \mathbf{u}^{k+1}), \quad (64)\\ \nabla_{\mathbf{u}} L(\mathbf{x}^{k+1}, \mathbf{z}^{k+1}, \mathbf{u}^{k+1}) &= \mu(\mathbf{A}\mathbf{x}^{k+1} - \mathbf{B}\mathbf{z}^{k+1}) = \mu(\mathbf{u}^{k+1} - \mathbf{u}^k). \quad (65)\end{aligned}$$

Here we use (15) and (17) for (63); (33) for (64); (17) for (65). By (33), Assumption 3.4(3), and Assumption 3.5:

$$\|\nabla_{\mathbf{x}} L(\mathbf{x}^{k+1}, \mathbf{z}^{k+1}, \mathbf{u}^{k+1})\| \le \sqrt{\rho(\mathbf{A}^T\mathbf{A})}(\mu + L_c)\|\mathbf{B}\mathbf{z}^{k+1} - \mathbf{B}\mathbf{z}^k\|,$$

$$\|\nabla_{\mathbf{z}} L(\mathbf{x}^{k+1}, \mathbf{z}^{k+1}, \mathbf{u}^{k+1})\| \le \sqrt{\rho(\mathbf{B}^T\mathbf{B})} L_c \|\mathbf{B}\mathbf{z}^{k+1} - \mathbf{B}\mathbf{z}^k\|.$$

And notice that $\nabla_{\mathbf{u}} L(\mathbf{x}^{k+1}, \mathbf{z}^{k+1}, \mathbf{u}^{k+1}) = -\mathbf{B}^T \nabla_{\mathbf{z}} L(\mathbf{x}^{k+1}, \mathbf{z}^{k+1}, \mathbf{u}^{k+1})$, then we get the result. □

F.2 Proof for Theorem 3.3

Recall that Theorem 3.3 is about linear convergence of the **x**-**z**-**u** iteration. To simplify the notation, we define:

$$\mathbf{N}(\mathbf{z}) := \mathbf{z} + \frac{1}{\mu}\mathbf{B}^{-T}\nabla g(\mathbf{B}^{-1}\mathbf{z}). \quad (66)$$

Proposition F.3. *The* **x**-**z**-**u** *iteration* (15)-(17) *satisfies*

$$\mathbf{N}(\mathbf{B}\mathbf{z}^{k+1}) = (\mathbf{I} + \mu\mathbf{K})^{-1}(\mathbf{A}\tilde{\mathbf{x}} + \mu\mathbf{K}\mathbf{B}z^k + \frac{1}{\mu}\mathbf{B}^{-T}\nabla g(\mathbf{z}^k)), \quad (67)$$

where matrix **K** *is defined in* (26).

Proof. By (17) we have:

$$\mathbf{B}\mathbf{z}^k - \mathbf{u}^k = \mathbf{A}\mathbf{x}^{k+1} + \mathbf{B}\mathbf{z}^k - \mathbf{B}\mathbf{z}^{k+1} - \mathbf{u}^{k+1}.$$

$$\overset{\text{by (15)}}{\Longrightarrow} (\mathbf{G} + \mu\mathbf{A}^T\mathbf{A})\mathbf{x}^{k+1} = \mathbf{G}\tilde{\mathbf{x}} + \mu\mathbf{A}^T(\mathbf{A}\mathbf{x}^{k+1} + \mathbf{B}\mathbf{z}^k - \mathbf{B}\mathbf{z}^{k+1} - \mathbf{u}^{k+1})$$

$$\Longrightarrow \mathbf{x}^{k+1} = \tilde{\mathbf{x}} + \mu\mathbf{G}^{-1}\mathbf{A}^T(\mathbf{B}\mathbf{z}^k - \mathbf{B}\mathbf{z}^{k+1} - \mathbf{u}^{k+1})$$

$$\Longrightarrow \mathbf{A}\mathbf{x}^{k+1} = \mathbf{A}\tilde{\mathbf{x}} + \mu\mathbf{A}\mathbf{G}^{-1}\mathbf{A}^T(\mathbf{B}\mathbf{z}^k - \mathbf{B}\mathbf{z}^{k+1} - \mathbf{u}^{k+1}).$$

$$\overset{\text{by (17)}}{\Longrightarrow} (\mathbf{I} + \mu\mathbf{A}\mathbf{G}^{-1}\mathbf{A}^T)(\mathbf{u}^{k+1} + \mathbf{B}\mathbf{z}^{k+1}) = \mathbf{A}\tilde{\mathbf{x}} + \mu\mathbf{A}\mathbf{G}^{-1}\mathbf{A}^T\mathbf{B}\mathbf{z}^k + \mathbf{u}^k.$$

$$\overset{\text{by (34)}}{\Longrightarrow} (\mathbf{I} + \mu\mathbf{A}\mathbf{G}^{-1}\mathbf{A}^T)(\frac{1}{\mu}\mathbf{B}^{-T}\nabla g(\mathbf{z}^{k+1}) + \mathbf{B}\mathbf{z}^{k+1})$$

$$= \mathbf{A}\tilde{\mathbf{x}} + \mu\mathbf{A}\mathbf{G}^{-1}\mathbf{A}^T\mathbf{B}\mathbf{z}^k + \frac{1}{\mu}\mathbf{B}^{-T}\nabla g(\mathbf{z}^k).$$

From the definitions of **N** and **K**, the last equation above becomes:

$$(\mathbf{I} + \mu\mathbf{K})\,\mathbf{N}(\mathbf{B}\mathbf{z}^{k+1}) = \mathbf{A}\tilde{\mathbf{x}} + \mu\mathbf{K}\mathbf{B}\mathbf{z}^k + \frac{1}{\mu}\mathbf{B}^{-T}\nabla g(\mathbf{z}^k)$$

$$\Longrightarrow \mathbf{N}(\mathbf{B}\mathbf{z}^{k+1}) = (\mathbf{I} + \mu\mathbf{K})^{-1}(\mathbf{A}\tilde{\mathbf{x}} + \mu\mathbf{K}\mathbf{B}z^k + \frac{1}{\mu}\mathbf{B}^{-T}\nabla g(\mathbf{z}^k)),$$

which completes the proof. □

Next we show a sufficient condition for the convergence to a stationary point:

Proposition F.4. *If the sequence* $\{\mathbf{z}^k\}$ *converges, then* $\{\mathbf{x}^k, \mathbf{z}^k, \mathbf{u}^k\}$ *converges to a stationary point defined in* (25).

Proof. Suppose $\mathbf{z}^k \to \mathbf{z}^*$. Then by (21), $\mathbf{u}^k \to \mathbf{u}^* = \mathbf{B}^{-T}\nabla g(\mathbf{z}^*)$, which proves $\nabla g(\mathbf{z}^*) - \mathbf{B}^T\mathbf{u}^* = 0$. By (15), $\mathbf{x}^k \to \mathbf{x}^*$ where

$$\mathbf{x}^* = (\mathbf{G} + \mu\mathbf{A}^T\mathbf{A})^{-1}(\mathbf{G}\tilde{\mathbf{x}} + \mu\mathbf{A}^T(\mathbf{B}\mathbf{z}^* + \mathbf{c} - \mathbf{u}^*)) \quad (68)$$

In (17), let $k \to \infty$ then we have

$$\mathbf{A}\mathbf{x}^* - \mathbf{B}\mathbf{z}^* = \mathbf{c} \quad (69)$$

The identity $\nabla f(\mathbf{x}^*) + \mathbf{A}^T\mathbf{u}^* = 0$ then follows from (68) and (69). □

We now show that $\{\mathbf{z}^k\}$ converge linearly:

Proof for Theorem 3.3. From (67):

$$\begin{aligned}&\mathbf{N}(\mathbf{B}\mathbf{z}^{k+1}) - \mathbf{N}(\mathbf{B}\mathbf{z}^k)\\ &= (\mathbf{I} + \mu\mathbf{K})^{-1}(\mu\mathbf{K}\mathbf{B}(\mathbf{z}^k - \mathbf{z}^{k-1}) + \frac{1}{\mu}\mathbf{B}^{-T}(\nabla g(\mathbf{z}^k) - \nabla g(\mathbf{z}^{k-1}))).\end{aligned} \quad (70)$$

By Proposition F.2, $g(\mathbf{z}^k) \le T(\mathbf{x}^0, \mathbf{z}^0) + c_1, \forall k \in \mathbb{N}$. Then by the definition of c_1(see Assumption 3.5) and Assumption 3.4(3):

$$\|\nabla\hat{g}(\mathbf{B}\mathbf{z}^{k+1}) - \nabla\hat{g}(\mathbf{B}\mathbf{z}^k)\| \le L_c\|\mathbf{B}\mathbf{z}^{k+1} - \mathbf{B}\mathbf{z}^k\|, \ \forall k \in \mathbb{N}$$

$$\Longrightarrow \|\mathbf{N}(\mathbf{B}\mathbf{z}^{k+1}) - \mathbf{N}(\mathbf{B}\mathbf{z}^k)\| \ge (1 - \frac{L_c}{\mu})\|\mathbf{B}\mathbf{z}^{k+1} - \mathbf{B}\mathbf{z}^k\|. \quad (71)$$

For the right hand side of (70):

$$\|(\mathbf{I} + \mu\mathbf{K})^{-1}(\mu\mathbf{K}\mathbf{B}(\mathbf{z}^k - \mathbf{z}^{k-1}) + \frac{1}{\mu}\mathbf{B}^{-T}(\nabla g(\mathbf{z}^k) - \nabla g(\mathbf{z}^{k-1})))\|$$

$$\le \|(\mathbf{I} + \mu\mathbf{K})^{-1}(\mu\mathbf{K}\mathbf{B}(\mathbf{z}^k - \mathbf{z}^{k-1})\| + \|\frac{1}{\mu}(\mathbf{I} + \mu\mathbf{K})^{-1}\mathbf{B}^{-T}(\nabla g(\mathbf{z}^k) - \nabla g(\mathbf{z}^{k-1}))\|.$$

By the spectral mapping theorem:

$$\|(\mathbf{I} + \mu\mathbf{K})^{-1}(\mu\mathbf{K})\| = \rho\left((\mathbf{I} + \mu\mathbf{K})^{-1}(\mu\mathbf{K})\right) = \frac{\mu\rho(\mathbf{K})}{1 + \mu\rho(\mathbf{K})}. \quad (72)$$

And notice that **K** is positive semi-definite:

$$\|\frac{1}{\mu}(\mathbf{I}+\mu\mathbf{K})^{-1}\mathbf{B}^{-T}(\nabla g(\mathbf{z}^k) - \nabla g(\mathbf{z}^{k-1})))\| \le \frac{L_c}{\mu}\|\mathbf{B}\mathbf{z}^k - \mathbf{B}\mathbf{z}^{k-1}\|. \quad (73)$$

Combine (72) with (73):

$$\|(\mathbf{I} + \mu\mathbf{K})^{-1}(\mu\mathbf{K}(\mathbf{B}\mathbf{z}^k - \mathbf{B}\mathbf{z}^{k-1}) + (\frac{1}{\mu}(\mathbf{B}^{-T}\nabla g(\mathbf{z}^k) - \mathbf{B}^{-T}\nabla g(\mathbf{z}^{k-1})))\|$$

$$\le (\frac{\mu\rho(\mathbf{K})}{1 + \mu\rho(\mathbf{K})} + \frac{L_c}{\mu})\|\mathbf{B}\mathbf{z}^k - \mathbf{B}\mathbf{z}^{k-1}\|. \quad (74)$$

By (71) and (74) we have:

$$\|\mathbf{B}\mathbf{z}^{k+1} - \mathbf{B}\mathbf{z}^k\| \le \frac{\frac{\mu\rho(\mathbf{K})}{1+\mu\rho(\mathbf{K})} + \frac{L_c}{\mu}}{1 - \frac{L_c}{\mu}}\|\mathbf{B}\mathbf{z}^k - \mathbf{B}\mathbf{z}^{k-1}\|. \quad (75)$$

If $\mu > \max\left\{\frac{1}{\frac{1}{2L_c} - \rho(\mathbf{K})}, \frac{1}{L_c}\right\}$ then $\gamma_1 < 1$, which completes the proof. □

F.3 Proof for Theorem 3.6

Theorem 3.6 is about general convergence of the **x**-**z**-**u** iteration. We first prove Proposition 3.5 that defines the value η.

Proof for Proposition 3.5. By the definition of $\mathbf{K}$ in (26), we know that $\mathbf{K}(R(\mathbf{A})) \subset R(\mathbf{A})$. Since $R(\mathbf{A})$ is a linear subspace and $\mathbf{K}$ is a linear operator, for the proof it suffices to show $\ker(\mathbf{K}) \cap R(\mathbf{A}) = \{0\}$, where $\ker(\mathbf{K})$ is the kernel of $\mathbf{K}$. Now assume $\mathbf{y} \in \ker(\mathbf{K})$, then for any $\mathbf{z} \in \mathbb{R}^q$ where q is the number of rows in matrix $\mathbf{A}$, we have:

$$\begin{aligned}\langle \mathbf{A}\mathbf{G}^{-1}\mathbf{A}^T\mathbf{y}, \mathbf{z}\rangle = 0 &\implies \langle \mathbf{G}^{-1}\mathbf{A}^T\mathbf{y}, \mathbf{A}^T\mathbf{z}\rangle = 0 \\ &\implies \langle \mathbf{G}^{-1}\mathbf{A}^T\mathbf{y}, \mathbf{A}^T\mathbf{y}\rangle = 0 \quad (\text{take } \mathbf{z} = \mathbf{y}).\end{aligned}$$

Notice that $\mathbf{G}^{-1}$ is positive definite, so we have $\mathbf{A}^T\mathbf{y} = 0$, which is equivalent to $\mathbf{y} \perp R(\mathbf{A})$. Hence we get $\ker(\mathbf{K}) \cap R(\mathbf{A}) = \{0\}$, which completes the proof. □

The next proposition provides a characterization of $\mathbf{u}^{k+1}$:

Proposition F.5. *The* **z**-**x**-**u** *iteration* (18)-(20) *satisfies:*

$$\mathbf{u}^{k+1} = \mathbf{A}\mathbf{x}^{k+1} - \mathbf{A}\mathbf{x}^k + \frac{1}{\mu}\mathbf{B}^{-T}\nabla g(\mathbf{z}^{k+1}). \quad (76)$$

Proof. From (20):

$$\mathbf{u}^{k+1} - \mathbf{A}\mathbf{x}^{k+1} = \mathbf{u}^k - \mathbf{B}\mathbf{z}^{k+1}. \quad (77)$$

From (23):

$$\begin{aligned}&\mathbf{A}\mathbf{x}^k + \mathbf{u}^k = \mathbf{A}\tilde{\mathbf{x}} - \mu\mathbf{K}\mathbf{u}^k + \mathbf{u}^k \\ \overset{\text{by (18)}}{\implies} &\mathbf{B}\mathbf{z}^{k+1} + \frac{1}{\mu}\mathbf{B}^{-T}\nabla g(\mathbf{z}^{k+1}) = \mathbf{A}\mathbf{x}^k + \mathbf{u}^k \\ \implies &\frac{1}{\mu}\mathbf{B}^{-T}\nabla g(\mathbf{z}^{k+1}) = \mathbf{A}\mathbf{x}^k + \mathbf{u}^k - \mathbf{B}\mathbf{z}^{k+1}. \quad (78)\end{aligned}$$

Combine (77) with (78) then we can get the result. □

Now we are able to bound both $\|\mathbf{u}^k\|$ and $\|\mathbf{u}^{k+1} - \mathbf{u}^k\|$:

Proposition F.6. *For* **z**-**x**-**u** *iteration* (18)-(20) *and* $k \geq 1$ *we have:*

$$\|\mathbf{u}^k\|^2 \leq \frac{4}{\eta^2\mu^2}\|\mathbf{A}\mathbf{x}^k - \mathbf{A}\tilde{\mathbf{x}}\|^2 + (\frac{4\rho(\mathbf{K})^2}{\mu^2\eta^2} + \frac{2}{\mu^2})\|\mathbf{B}^{-T}\nabla g(\mathbf{z}^k)\|^2, \quad (79)$$

$$\begin{aligned}\|\mathbf{u}^{k+1} - \mathbf{u}^k\|^2 \leq &\frac{4}{\mu^2\eta^2}\|\mathbf{A}\mathbf{x}^{k+1} - \mathbf{A}\mathbf{x}^k\|^2 \\ &+ (\frac{4\rho(\mathbf{K})^2}{\mu^2\eta^2} + \frac{2}{\mu^2})\|\mathbf{B}^{-T}(\nabla g(\mathbf{z}^{k+1}) - \nabla g(\mathbf{z}^k))\|^2. \quad (80)\end{aligned}$$

Proof. To prove (79), note that from (76):

$$\|\mathbf{u}^k\|^2 \leq 2\|\mathbf{A}\mathbf{x}^k - \mathbf{A}\mathbf{x}^{k-1}\|^2 + \frac{2}{\mu^2}\|\mathbf{B}^{-T}\nabla g(\mathbf{z}^k)\|^2. \quad (81)$$

And from (23):

$$\begin{aligned}&\mathbf{A}\mathbf{x}^{k+1} = \mathbf{A}\tilde{\mathbf{x}} - \mu\mathbf{K}\mathbf{u}^{k+1} \quad (82) \\ \overset{\text{by (76)}}{\implies} &\mathbf{A}\mathbf{x}^{k+1} = \mathbf{A}\tilde{\mathbf{x}} - \mu\mathbf{K}(\mathbf{A}\mathbf{x}^{k+1} - \mathbf{A}\mathbf{x}^k) - \mathbf{K}\mathbf{B}^{-T}\nabla g(\mathbf{z}^{k+1}) \\ \implies &\mathbf{A}\mathbf{x}^{k+1} - \mathbf{A}\tilde{\mathbf{x}} + \mathbf{K}\mathbf{B}^{-T}\nabla g(\mathbf{z}^{k+1}) = -\mu\mathbf{K}(\mathbf{A}\mathbf{x}^{k+1} - \mathbf{A}\mathbf{x}^k). \quad (83)\end{aligned}$$

Hence by Proposition 3.5:

$$\mu^2\eta^2\|\mathbf{A}\mathbf{x}^{k+1} - \mathbf{A}\mathbf{x}^k\|^2 \leq 2\|\mathbf{A}\mathbf{x}^{k+1} - \mathbf{A}\tilde{\mathbf{x}}\|^2 + 2\rho(\mathbf{K})^2\|\mathbf{B}^{-T}\nabla g(\mathbf{z}^{k+1})\|^2,$$

and (79) follows from this equation and (81). For (80), from (76):

$$\begin{aligned}&\mathbf{u}^{k+1} - \mathbf{u}^k = \mathbf{A}(\mathbf{x}^{k+1} - 2\mathbf{x}^k + \mathbf{x}^{k-1}) + \frac{1}{\mu}\mathbf{B}^{-T}(\nabla g(\mathbf{z}^{k+1}) - \nabla g(\mathbf{z}^k)) \\ \implies &\|\mathbf{u}^{k+1} - \mathbf{u}^k\|^2 \leq 2\|\mathbf{A}(\mathbf{x}^{k+1} - 2\mathbf{x}^k + \mathbf{x}^{k-1})\|^2 \\ &\qquad + \frac{2}{\mu^2}\|\mathbf{B}^{-T}(\nabla g(\mathbf{z}^{k+1}) - \nabla g(\mathbf{z}^k))\|^2. \quad (84)\end{aligned}$$

And by (83):

$$\mathbf{A}\mathbf{x}^{k+1} - \mathbf{A}\mathbf{x}^k + \mathbf{K}\mathbf{B}^{-T}(\nabla g(\mathbf{z}^{k+1}) - \nabla g(\mathbf{z}^k)) = -\mu\mathbf{K}\mathbf{A}(\mathbf{x}^{k+1} - 2\mathbf{x}^k + \mathbf{x}^{k-1}).$$

Hence:

$$\begin{aligned}&\mu^2\eta^2\|\mathbf{A}(\mathbf{x}^{k+1} - 2\mathbf{x}^k + \mathbf{x}^{k-1})\|^2 \\ \leq\; &2\|\mathbf{A}\mathbf{x}^{k+1} - \mathbf{A}\mathbf{x}^k\|^2 + 2\rho(\mathbf{K})^2\|\mathbf{B}^{-T}(\nabla g(\mathbf{z}^{k+1}) - \nabla g(\mathbf{z}^k))\|^2. \end{aligned} \quad (85)$$

Then (80) follows from (84) and (85). □

Similar to Proposition F.2, we can prove:

Proposition F.7. *Suppose Assumptions 3.4 and 3.6 hold, and μ is sufficiently large. The the* **z**-**x**-**u** *iteration satisfies:*

$$T(\mathbf{x}^k, \mathbf{z}^k) \leq T(\mathbf{x}^0, \mathbf{z}^0) + c_2 + c_3, \; L(\mathbf{x}^k, \mathbf{z}^k, \mathbf{u}^k) \leq T(\mathbf{x}^0, \mathbf{z}^0) + c_3. \quad (86)$$

Proof. We will prove this by induction. For $k = 0$ this is trivial, now assume (86) holds for every $k \leq l$. Consider $k = l + 1$. By the definition of $\mathbf{z}^{l+1}$ in (18):

$$g(\mathbf{z}^{l+1}) + \frac{\mu}{2}\|\mathbf{A}\mathbf{x}^l - \mathbf{B}\mathbf{z}^{l+1} + \mathbf{u}^l\|^2 \leq g(\mathbf{z}^l) + \frac{\mu}{2}\|\mathbf{A}\mathbf{x}^l - \mathbf{B}\mathbf{z}^l + \mathbf{u}^l\|^2. \quad (87)$$

By the definition of $\mathbf{x}^{l+1}$ in (18):

$$f(\mathbf{x}^{l+1}) + \frac{\mu}{2}\|\mathbf{A}\mathbf{x}^{l+1} - \mathbf{B}\mathbf{z}^{l+1} + \mathbf{u}^l\|^2 \leq f(\mathbf{x}^l) + \frac{\mu}{2}\|\mathbf{A}\mathbf{x}^l - \mathbf{B}\mathbf{z}^{l+1} + \mathbf{u}^l\|^2. \quad (88)$$

add (88) to (87):

$$T(\mathbf{x}^{l+1}, \mathbf{z}^{l+1}) \leq L(\mathbf{x}^l, \mathbf{z}^l, \mathbf{u}^l) + \frac{\mu}{2}\|\mathbf{u}^l\|^2.$$

By induction:

$$L(\mathbf{x}^l, \mathbf{z}^l, \mathbf{u}^l) \leq T(\mathbf{x}^0, \mathbf{z}^0) + c_3.$$

Since $l + 1 \geq 1$, by Proposition F.6:

$$\frac{\mu}{2}\|\mathbf{u}^l\|^2 \leq \frac{2}{\eta^2\mu}\|\mathbf{A}\mathbf{x}^l - \mathbf{A}\tilde{\mathbf{x}}\|^2 + (\frac{2\rho(\mathbf{K})^2}{\mu\eta^2} + \frac{1}{\mu})\|\mathbf{B}^{-T}\nabla g(\mathbf{z}^l)\|^2.$$

By induction:

$$T(\mathbf{x}^l, \mathbf{z}^l) \leq T(\mathbf{x}^0, \mathbf{z}^0) + c_2 + c_3 \leq T(\mathbf{x}^0, \mathbf{z}^0) + 1.$$

By the definition of c_2, $\frac{\mu}{2}\|\mathbf{u}^l\|^2 \leq c_2$. Hence:

$$T(\mathbf{x}^{l+1}, \mathbf{z}^{l+1}) \leq T(\mathbf{x}^0, \mathbf{z}^0) + c_2 + c_3,$$

which proves the first part. For the second part, we first prove that the conclusion holds for $l = 0$ ($k = 1$). From the first part we know:

$$T(\mathbf{x}^1, \mathbf{z}^1) \leq T(\mathbf{x}^0, \mathbf{z}^0) + c_2 + c_3.$$

Notice that $f(\mathbf{x}^1) \geq 0$ so we have $g(\mathbf{z}^1) \leq T(\mathbf{x}^0, \mathbf{z}^0) + c_2 + c_3$. Hence by Proposition F.1:

$$L(\mathbf{x}^0, \mathbf{z}^1, \mathbf{u}^0) \leq L(\mathbf{x}^0, \mathbf{z}^0, \mathbf{u}^0) - \frac{\mu - L_d}{2}\|\mathbf{B}\mathbf{z}^1 - \mathbf{B}\mathbf{z}^0\|^2.$$

And by Assumption 3.2:

$$L(\mathbf{x}^1, \mathbf{z}^1, \mathbf{u}^0) \leq L(\mathbf{x}^0, \mathbf{z}^1, \mathbf{u}^0) - \frac{\mu}{2}\|\mathbf{A}\mathbf{x}^1 - \mathbf{A}\mathbf{x}^0\|^2.$$

Moreover, we have:

$$L(\mathbf{x}^1, \mathbf{z}^1, \mathbf{u}^1) = L(\mathbf{x}^1, \mathbf{z}^1, \mathbf{u}^0) + \mu\|\mathbf{u}^1 - \mathbf{u}^0\|^2$$

By (79) and $\mathbf{u}^0 = 0$:

$$\begin{aligned}\mu\|\mathbf{u}^1 - \mathbf{u}^0\|^2 &= \mu\|\mathbf{u}^1\|^2 \\ &= \frac{4}{\eta^2\mu}\|\mathbf{A}\mathbf{x}^1 - \mathbf{A}\tilde{\mathbf{x}}\|^2 + (\frac{4\rho(\mathbf{K})^2}{\mu\eta^2} + \frac{2}{\mu})\|\mathbf{B}^{-T}\nabla g(\mathbf{z}^1)\|^2.\end{aligned}$$

Moreover, we have:

$$\begin{aligned}\|\mathbf{B}^{-T}\nabla g(\mathbf{z}^1)\|^2 &\le 2\|\mathbf{B}^{-T}\nabla g(\mathbf{z}^1) - \mathbf{B}^{-T}\nabla g(\mathbf{z}^0)\|^2 + 2\|\mathbf{B}^{-T}\nabla g(\mathbf{z}^0)\|^2 \\ &\le 2L_d\|\mathbf{B}\mathbf{z}^1 - \mathbf{B}\mathbf{z}^0\|^2 + 2\|\mathbf{B}^{-T}\nabla g(\mathbf{z}^0)\|^2.\end{aligned}$$

So if $\frac{\mu}{2} \ge \frac{4}{\eta^2\mu}$ and $\frac{\mu - L_d}{2} \ge 2L_d(\frac{4\rho(\mathbf{K})^2}{\mu\eta^2} + \frac{2}{\mu})$, then we have:

$$\begin{aligned}L(\mathbf{x}^1, \mathbf{z}^1, \mathbf{u}^1) &\le L(\mathbf{x}^0, \mathbf{z}^0, \mathbf{u}^0) + (\frac{8\rho(\mathbf{K})^2}{\mu\eta^2} + \frac{4}{\mu})\|\mathbf{B}^{-T}\nabla g(\mathbf{z}^0)\|^2 \\ &= T(\mathbf{x}^0, \mathbf{z}^0) + (\frac{8\rho(\mathbf{K})^2}{\mu\eta^2} + \frac{4}{\mu})\|\mathbf{B}^{-T}\nabla g(\mathbf{z}^0)\|^2.\end{aligned}$$

By the definition of c_3 we have $L(\mathbf{x}^1, \mathbf{z}^1, \mathbf{u}^1) \le T(\mathbf{x}^0, \mathbf{z}^0) + c_3$. Now suppose $l \ge 1$. Similar to the proof of the case $l = 0$ we have:

$$\begin{aligned}L(\mathbf{x}^l, \mathbf{z}^{l+1}, \mathbf{u}^l) &\le L(\mathbf{x}^l, \mathbf{z}^l, \mathbf{u}^l) - \frac{\mu - L_d}{2}\|\mathbf{B}\mathbf{z}^{l+1} - \mathbf{B}\mathbf{z}^l\|^2, \\ L(\mathbf{x}^{l+1}, \mathbf{z}^{l+1}, \mathbf{u}^l) &\le L(\mathbf{x}^l, \mathbf{z}^{l+1}, \mathbf{u}^l) - \frac{\mu}{2}\|\mathbf{A}\mathbf{x}^{l+1} - \mathbf{A}\mathbf{x}^l\|^2, \\ L(\mathbf{x}^{l+1}, \mathbf{z}^{l+1}, \mathbf{u}^{l+1}) &= L(\mathbf{x}^{l+1}, \mathbf{z}^{l+1}, \mathbf{u}^l) + \mu\|\mathbf{u}^{l+1} - \mathbf{u}^l\|^2.\end{aligned}$$

By (80) we have:

$$\begin{aligned}\mu\|\mathbf{u}^{l+1} - \mathbf{u}^l\|^2 \le &\frac{4}{\mu\eta^2}\|\mathbf{A}\mathbf{x}^{l+1} - \mathbf{A}\mathbf{x}^l\|^2 \\ &+ (\frac{4\rho(\mathbf{K})^2}{\mu\eta^2} + \frac{2}{\mu})\|\mathbf{B}^{-T}(\nabla g(\mathbf{z}^{l+1}) - \nabla g(\mathbf{z}^l))\|^2.\end{aligned}$$

Since $g(\mathbf{z}^l), g(\mathbf{z}^{l+1}) \le T(\mathbf{x}^0, \mathbf{z}^0) + c_2 + c_3$, by the definition of L_d:

$$\|\mathbf{B}^{-T}(\nabla g(\mathbf{z}^{l+1}) - \nabla g(\mathbf{z}^l))\| \le L_d\|\mathbf{B}\mathbf{z}^{l+1} - \mathbf{B}\mathbf{z}^l\|. \quad (89)$$

Hence we have:

$$\begin{aligned}\mu\|\mathbf{u}^{l+1} - \mathbf{u}^l\|^2 \le &\frac{4}{\mu\eta^2}\|\mathbf{A}\mathbf{x}^{l+1} - \mathbf{A}\mathbf{x}^l\|^2 \\ &+ (\frac{4\rho(\mathbf{K})^2L_d^2}{\mu\eta^2} + \frac{2L_d^2}{\mu})\|\mathbf{B}\mathbf{z}^{l+1} - \mathbf{B}\mathbf{z}^l\|^2.\end{aligned}$$

If $\frac{\mu}{2} \ge \frac{4}{\eta^2\mu}$ and $\frac{\mu - L_d}{2} \ge (\frac{4\rho(\mathbf{K})^2L_d^2}{\mu\eta^2} + \frac{2L_d^2}{\mu})$, then we have:

$$L(\mathbf{x}^{l+1}, \mathbf{z}^{l+1}, \mathbf{u}^{l+1}) \le L(\mathbf{x}^l, \mathbf{z}^l, \mathbf{u}^l) \le T(\mathbf{x}^0, \mathbf{z}^0) + c_3 \quad (90)$$

which completes the proof. □

Similar to the proof of Theorem 3.5, we need to show (P1)-(P4) hold for **z-x-u** iteration. Sufficient descent has already been shown in the proof of Proposition F.7. The remaining part is the same as the proof of Theorem 3.5 so we omit it.

F.4 Proof for Theorem 3.4

Theorem 3.4 is about linear convergence of the **z-x-u** iteration. Similar to Proposition F.4, for the convergence of the **z-x-u** iteration to a stationary point, it suffices to show that the sequence $\{\mathbf{u}^k\}$ converges. Then for the main proof:

Proof for Theorem 3.4. By (78):

$$\begin{aligned}&\mathbf{B}\mathbf{z}^{k+1} + \frac{1}{\mu}\mathbf{B}^{-T}\nabla g(\mathbf{z}^{k+1}) = \mathbf{A}\tilde{\mathbf{x}} - \mathbf{v}^k \\ \Longrightarrow\ &\mathbf{B}(\mathbf{z}^{k+1} - \mathbf{z}^k) + \frac{1}{\mu}\mathbf{B}^{-T}(\nabla g(\mathbf{z}^{k+1}) - \nabla g(\mathbf{z}^k)) = -(\mathbf{v}^k - \mathbf{v}^{k-1}).\end{aligned}$$

By (89):

$$(1 - \frac{L_d}{\mu})\|\mathbf{B}\mathbf{z}^{k+1} - \mathbf{B}\mathbf{z}^k\| \le \|\mathbf{v}^k - \mathbf{v}^{k-1}\|.$$

Hence we have:

$$\begin{aligned}\frac{1}{\mu}\|\mathbf{B}^{-T}(\nabla g(\mathbf{z}^{k+1}) - \nabla g(\mathbf{z}^k))\| &\le \frac{L_d}{\mu}\|\mathbf{B}\mathbf{z}^{k+1} - \mathbf{B}\mathbf{z}^k\| \\ &\le \frac{L_d}{\mu - L_d}\|\mathbf{v}^k - \mathbf{v}^{k-1}\|. \quad (91)\end{aligned}$$

By (76) and (82):

$$\begin{aligned}&(\mathbf{I} + \mu\mathbf{K})\mathbf{u}^{k+1} = \mu\mathbf{K}\mathbf{u}^k + \frac{1}{\mu}\mathbf{B}^{-T}\nabla g(\mathbf{z}^{k+1}) \\ \Longrightarrow\ &\mathbf{v}^{k+1} = (\mathbf{I} + \mu\mathbf{K})^{-1}\mu\mathbf{K}\mathbf{v}^k + (\mathbf{I} + \mu\mathbf{K})^{-1}(\mathbf{I} - \mu\mathbf{K})\frac{1}{\mu}\mathbf{B}^{-T}\nabla g(\mathbf{z}^{k+1}).\end{aligned}$$

Hence we have:

$$\begin{aligned}\|\mathbf{v}^{k+1} - \mathbf{v}^k\| \le &\|(\mathbf{I} + \mu\mathbf{K})^{-1}\mu\mathbf{K}(\mathbf{v}^k - \mathbf{v}^{k-1})\| \\ &+ \frac{1}{\mu}\|(\mathbf{I} + \mu\mathbf{K})^{-1}(\mathbf{I} - \mu\mathbf{K})\mathbf{B}^{-T}(\nabla g(\mathbf{z}^{k+1}) - \nabla g(\mathbf{z}^k))\|.\end{aligned}$$

Similar to (72) we have:

$$\begin{aligned}&\|(\mathbf{I} + \mu\mathbf{K})^{-1}\mu\mathbf{K}(\mathbf{v}^k - \mathbf{v}^{k-1})\| \le \frac{\mu\rho(\mathbf{K})}{1 + \mu\rho(\mathbf{K})}\|\mathbf{v}^k - \mathbf{v}^{k-1}\| \\ \overset{\text{by (91)}}{\Longrightarrow}\ &\|\mathbf{v}^{k+1} - \mathbf{v}^k\| \le (\frac{\mu\rho(\mathbf{K})}{1 + \mu\rho(\mathbf{K})} + \frac{L_d}{\mu - L_d})\|\mathbf{v}^k - \mathbf{v}^{k-1}\|.\end{aligned}$$

Then let $\mu > \max\left\{\frac{2}{\frac{1}{L_d} - \rho(\mathbf{K})}, \frac{1}{L_d}\right\}$ and we get the result. □

F.5 Sketch of proofs for Theorems 3.1 and 3.2

For the proof of Theorem 3.1, the derivation can start from (71) without assumptions on the initial values. The rest of the proofs is the same as the proof of Theorem 3.3.

For the proof of Theorem 3.2, the derivation of (89) does not rely on the initial value. The rest is the same.