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

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.

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; Koalsky 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. [2007] proposed 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] 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


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 and Wohlberg 2015; Mišik et al. 2014; Lai and Osher 2014; Liavas and Sidirooulos 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 Fong 2015; Hong et al. 2016; Magnusson 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 Kadkhodaei 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

\[
\min_{x} \phi(x, Dx + h). \tag{1}
\]

Here \(x\) 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 \(Dx + h\) encodes a transformation of the positions \(x\) relevant for the optimization problem, such as the deformation gradient of each tetrahedron element in an elastic object. The notation \(\phi(x, Dx + h)\) signifies that the target function contains a term that directly depends on \(Dx + h\), such as elastic energy dependent on the deformation gradient. In some applications, the optimization enforces hard constraints on \(x\) or \(Dx + h\), 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 \(y \in C\) where \(y\) is a subset from the components of \(x\) or \(Dx + h\), and \(C\) is the feasible
set. Then we include the following term into $\Phi$:

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

By definition, if $x^*$ is a solution, then the corresponding components $y^*$ must satisfy $y^* \in C$; 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 non-linear, 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 $z = Dx + h$ to derive an equivalent problem

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

where $W$ is a diagonal matrix with positive diagonal elements. $W$ 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

$$
\min_{x,z} \Phi(x, z) \quad \text{s.t.} \ Ax - Bz = c,
$$

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

$$
L(x, z, u) = \Phi(x, z) + (\mu u, Ax - Bz - c) + \frac{\mu}{2} \|Ax - Bz - c\|^2
= \Phi(x, z) + \frac{\mu}{2} \|Ax - Bz - c\|^2 - \frac{\mu}{2} \|u\|^2.
$$

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

$$
x^{k+1} = \arg\min_x L(x, z^k, u^k),
$$

$$
z^{k+1} = \arg\min_z L(x^{k+1}, z, u^k),
$$

$$
u^{k+1} = u^k + Ax^{k+1} - Bz^{k+1} - c.
$$

We can also update $z$ before $x$, resulting in an alternative scheme:

$$
z^{k+1} = \arg\min_z L(x^k, z, u^k),
$$

$$
x^{k+1} = \arg\min_x L(x, z^{k+1}, u^k),
$$

$$
u^{k+1} = u^k + Ax^{k+1} - Bz^{k+1} - c.
$$

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 $x$ and $z$ 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

$$
r_p^{k+1} = Ax^{k+1} - Bz^{k+1} - c.
$$

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

$$
r_d^{k+1} = \mu A^T B (x^{k+1} - z^k),
$$

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

$$
r_d^{k+1} = \mu B^T A (x^{k+1} - x^k).
$$

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 $\|r_p^{k+1}\|$ and $\|r_d^{k+1}\|$ are small enough.

**Anderson acceleration.** ADMM is easy to parallelize and converges 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 $q^{k-m}, q^{k-m+1}, \ldots, q^k$ 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 $F^j = G(q^j) - q^j$ ($j = k - m, \ldots, k$). Then the accelerated iterate is computed as

$$
q_{AA}^{k+1} = (1 - \beta) \left( q^k - \sum_{j=1}^m \theta_j^* (q^{k-j+1} - q^{k-j}) \right)
+ \beta \left( G(q^k) - \sum_{j=1}^m \theta_j^* (G(q^{k-j+1}) - G(q^{k-j})) \right),
$$

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

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

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 $x^{k+1}$ is dependent only on $z^k$ and $u^k$. Therefore, by treating $x^{k+1}$ as a function of $(z^k, u^k)$, we can rewrite $\theta_{k+1}$, and subsequently $u^{k+1}$, as a function of $(z^k, u^k)$ as well. In this way, the $x$-$z$-$u$ iteration can be treated as a fixed-point iteration of $(z, u)$:

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

Similarly, we can treat the $z$-$x$-$u$ scheme as a fixed-point iteration of $(x, u)$. 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

$$
r_{x-z-u}^{k+1} = \mu \|Ax^{k+1} - Bz^{k+1} - c\|^2 + \mu \|B(x^{k+1} - z^k)\|^2.
$$

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 z-x-u iteration.

Data: \( x^0, z^0, u^0 \): initial values of variables; 
\( L \): the augmented Lagrangian function; 
\( m \): the number of previous iterates used for acceleration; 
\( AA(G, F) \): Anderson acceleration from a sequence \( G \) of fixed-point mapping results of previous iterates, and a sequence \( F \) of their corresponding fixed-point residuals; 
\( l_{\text{max}} \): the maximum number of iterations; 
\( \epsilon \): convergence threshold for combined residual.

1. \( x_{\text{default}} = x^0; z_{\text{default}} = z^0; u_{\text{default}} = u^0;\)
2. \( \rho_{\text{prev}} = +\infty; j = 0; \text{reset} = \text{TRUE}; k = 0;\)
3. while TRUE do
   // Run one iteration of ADMM
   1. \( x_* = \text{argmin}_x L(x, z^k, u^k);\)
   2. \( z_* = \text{argmin}_z L(x_*, z, u^k);\)
   3. \( u_* = u^k + Ax_* - Bz_* - c;\)
   // Compute the combined residual
   4. \( r = \|Ax_* - Bz_* - c\|^2 + \|B(z_* - x^k)\|^2;\)
   5. if reset = TRUE OR \( r < \rho_{\text{prev}} \) then
      // Record the latest accepted iterate
      1. \( x_{\text{default}} = x_*; z_{\text{default}} = z_*; u_{\text{default}} = u_*;\)
      2. \( \rho_{\text{prev}} = r; \text{reset} = \text{FALSE};\)
      // Compute the accelerated iterate
      3. \( g_j = (x_* - z_*); f_j = (x_* - z^k, u_* - u^k);\)
      4. \( j = j + 1; m = \text{min}(m - 1, j);\)
      5. \( (x^{k+1}, u^{k+1}) = AA([g_j, \ldots, g_{j-m}], [f_j, \ldots, f_{j-m}]);\)
      6. \( k = k + 1;\)
   else
   // Revert to the last accepted iterate
   1. \( x^k = \text{default}; u^k = u_{\text{default}}; \text{reset} = \text{TRUE};\)
   end if
   7. if \( k \geq l_{\text{max}} \) OR \( r < \epsilon \) then
      // Check termination
      1. return \( x_{\text{default}};\)
   end if
4. end while

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

1. \( x_{\text{default}} = x^0; u_{\text{default}} = u^0; r_{\text{prev}} = +\infty; j = 0; \text{reset} = \text{TRUE}; k = 0;\)
2. while TRUE do
   1. \( z_* = \text{argmin}_z L(x^k, z, u^k);\)
   2. \( x_* = \text{argmin}_x L(x, z_*, u^k);\)
   3. \( u_* = u^k + Ax_* - Bz_* - c;\)
   4. \( r = \|Ax_* - Bz_* - c\|^2 + \|A(x_* - x^k)\|^2;\)
   5. if reset = TRUE OR \( r < r_{\text{prev}} \) then
      1. \( x_{\text{default}} = x_*; u_{\text{default}} = u_*; r_{\text{prev}} = r; \text{reset} = \text{FALSE};\)
      2. \( j = j + 1; m = \text{min}(m - 1, j);\)
      3. \( g_j = (x_* - u_*); f_j = (x_* - x^k, u_* - u^k);\)
      4. \( (x^{k+1}, u^{k+1}) = AA([g_j, \ldots, g_{j-m}], [f_j, \ldots, f_{j-m}]);\)
      5. \( k = k + 1;\)
   else
   // Revert to the last accepted iterate
   1. \( x^k = \text{default}; u^k = u_{\text{default}}; \text{reset} = \text{TRUE};\)
   end if
   1. end if
4. end while

The combined residual for the z-x-u iteration is defined as
\[
\rho^{k+1}_{z-x-u} = \|Ax^{k+1} - Bz^{k+1} - c\|^2 + \|A(x^{k+1} - x^k)\|^2.
\] (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 \( Ax^{k+1} - Bz^{k+1} - c \) and the variable changes \( x^{k+1} - x^k \) and \( z^{k+1} - x^k \) 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 \((z_{\text{AA}}, u_{\text{AA}})\), it is often difficult to find a pair \((z_i, u_i)\) that leads to a combined residual after one ADMM iteration (i.e., \((z_{\text{AA}}, u_{\text{AA}}) = G(z_i, u_i)\)). Therefore, we run one ADMM iteration on \((z_{\text{AA}}, u_{\text{AA}})\) instead, and use the resulting values \((z_*, u_*) = G(z_{\text{AA}}, u_{\text{AA}})\) to evaluate the combined residual. If the accelerated iterate is accepted, then the computation of \((z_*, u_*)\) 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 \( \Phi \) 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 \( z-x-u \) iteration, or \( x \) in the \( x-z-u \) 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 \( x_{\text{AA}} \) or \( z_{\text{AA}} \) 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 \( \Phi \) 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
\[
\min_{x,z} f(x) + g(z), \quad \text{s.t.} \quad Ax - Bz = c.
\] (13)
Moreover, we assume this problem satisfies the following properties:

ACM Trans. Graph., Vol. 38, No. 6, Article 163. Publication date: November 2019.
ASSUMPTION 3.1. Matrix $B$ is invertible.

ASSUMPTION 3.2. $f(x)$ is a strongly convex quadratic function

$$f(x) = \frac{1}{2}(x - \hat{x})^T G(x - \hat{x}),$$

where $\hat{x}$ is a constant and $G$ 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 $\hat{x}$ is the predicted value of node positions $x$ without internal forces, $G = M/\Delta t^2$ where $M$ is the mass matrix and $\Delta t$ is the integration time step, the auxiliary variable $z$ stacks the deformation gradient of each element, and $g(z)$ sums the elastic potential energy for all elements. For the problem (13), the $x$-$z$-$u$ iteration of ADMM becomes

$$x^{k+1} = (G + \mu A^T A)^{-1}(G\hat{x} + \mu A^T (Bz^k + c - u^k)),$$

$$z^{k+1} = \arg\min_{z} (g(z) + \frac{\mu}{2}||Ax^{k+1} - Bz - c + u^k||^2),$$

$$u^{k+1} = u^k + Ax^{k+1} - Bz^{k+1} - c.$$  

And the $x$-$z$-$u$ iteration becomes

$$z^{k+1} = \arg\min_{z} (g(z) + \frac{\mu}{2}||Ax^k - Bz - c + u^k||^2),$$

$$x^{k+1} = (G + \mu A^T A)^{-1}(G\hat{x} + \mu A^T (Bz^{k+1} + c - u^k)),$$

$$u^{k+1} = u^k + Ax^{k+1} - Bz^{k+1} - c.$$  

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 $(z, u)$ or $(x, u)$. 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 $x$ and $u$, it always returns the same value of $z$.

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

PROPOSITION 3.1. If the optimization problem (13) satisfies Assumptions 3.1 and 3.2, and the function $g(z)$ is differentiable, then the $x$-$z$-$u$ iteration (15)-(17) satisfies

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

A proof is given in Appendix A. Proposition 3.1 shows that $u^{k+1}$ can be recovered from $z^{k+1}$. Therefore, we can treat the $x$-$z$-$u$ iteration (15)-(17) as a fixed-point iteration of $z$ instead of $(x, u)$, and apply Anderson acceleration to $z$ alone. From the accelerated $z^{\star\star}$, we recover its corresponding dual variable $u^{\star\star}$ 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 $B$ is invertible, $u$ and $z$ 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:

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

$$Ax^{k+2} - Bz^{k+1} - c = 0.$$  

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 $x$ in the next iteration. Accordingly, we evaluate the effectiveness of an accelerated iterate $z_{\star\star}$ and its corresponding dual variable $u_{\star\star}$ by first computing a new value $x_{\star}$ according to the $x$-update step (15), then evaluating a residual $r_{xu} = Ax_{\star} - Bz_{\star\star} - c$. We only accept $z_{\star\star}$ 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 $z_{\star\star}$ is accepted, then $x_{\star}$ 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 \( z_{AA} \), 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 \( z \) alone leads to a faster decrease of combined residual than accelerating \( z, u \) 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

\[
x^{k+1} = \hat{x} - \mu G^{-1} A^T u^{k+1}.
\]

(23)

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

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

\[
Ax^{k+1} - Bz^{k+2} - c = 0.
\]

(24)

Accordingly, we evaluate the effectiveness of \( u_{AA} \) and \( x_{AA} \) by computing from them a \( z_{AA} \) using Eq. (18), and evaluating the residual \( r_{z-x-u} = Ax_{AA} - Bz_{AA} - c \). We accept \( u_{AA} \) if the norm of this residual is smaller than the previous iteration, and revert to the last un-accelerated iterate otherwise. If \( u_{AA} \) is accepted, then \( z_{AA} \) 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.

### 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; Magnusson 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 \( \mathbb{R} \cup \{+\infty\} \). Following [Rockafellar 1997], for a function

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

1. \( t_{prev} = +\infty \); \( j = 0 \); \( reset = TRUE \); \( k = 0 \).
2. while \( TRUE \) do
   1. Update \( z \) with (18) and compute residual with (24)
   2. \( z^{k+1} = \min_{z} \{g(z) + \frac{\mu}{2} \|Ax^k - Bz - c + u^k\|^2\} \)
   3. \( r = \|Ax^k - Bz^{k+1} - c\| \);
   4. Check whether the residual increases
   5. if \( reset == FALSE \) AND \( r >= r_{prev} \) then
      1. Revert to un-accelerated \( x, u \)
      2. \( x^k = x_{default} \); \( u^k = u_{default} \);
      3. \( z^{k+1} = \min_{z} \{g(z) + \frac{\mu}{2} \|Ax^k - Bz - c + u^k\|^2\} \);
      4. \( r = \|Ax^k - Bz^{k+1} - c\| \);
      5. \( reset = TRUE \);
   6. end if
   7. if \( k + 1 >= l_{max} \) OR \( r < \epsilon \) then
      1. return \( x^k \);
   8. end if
3. if Compute un-accelerated \( x \) and \( u \)
   9. \( x_{default} = (G + \mu A^T A)^{-1}(Gx + \mu A^T(Bz^{k+1} + c - u^k)) \);
   10. \( u_{default} = u^k + Ax_{default} - Bz^{k+1} - c \);
4. Use history of \( u \) to compute affine coefficients
   11. \( j = j + 1 \); \( m = \min(m, j) \);
   12. \( g^k_j = x_{default} \); \( g^k_j = u_{default} - g^k \);
   13. \( \theta^j_1, \ldots, \theta^j_m = \arg\min_{\theta^j_1, \ldots, \theta^j_m} \| \sum_{i=1}^m \theta_i f_i^m - f_i^j \|^2 \);
5. Compute accelerated \( x \) and \( u \) with the coefficients
   14. \( x^{k+1} = g^k_1 - \sum_{i=1}^m \theta_i^j f_i^{k+1} - g^k \);
   15. \( u^{k+1} = g^k_1 - \sum_{i=1}^m \theta_i^j f_i^{k+1} - g^k \);
   16. \( k = k + 1 \); \( t_{prev} = r \);
   17. end while
We denote by \( \mathcal{L}_F^\alpha \) a level-bounded set for any \( \alpha \in \mathbb{R} \). A function \( F \) is level-bounded if \( \mathcal{L}_F^\alpha \) is a bounded set for any \( \alpha \in \mathbb{R} \). Given a set \( S \), let \( S_0 \) and \( B_S \) denote the interior and the boundary of \( S \), respectively. A function \( F \) is continuous on \( \mathbb{R}^n \) if: (i) it is continuous within \( I_{\text{dom}(F)} \) in the conventional sense; and (ii) \( \forall x_k \rightarrow x \in \mathbb{R}^n \), we have \( F(x_k) \rightarrow F(x) = -\infty \). We say a function is Lipschitz continuous if it is differentiable and its gradient is Lipschitz continuous. Unless specified otherwise, \( \text{dom}(F) \) denotes the identity matrix and the identity map. The symbol \( \text{conv}(S) \) denotes the convex hull of a set \( S \), and \( \partial F \) denotes the set of all sub-differentials for a function \( F \) (see Rockafellar and Wets 2009, Definition 8.3(b)). For matrix \( Q \), we use \( \rho(Q) \) to represent its spectral radius. We will discuss conditions for the ADMM iterates \((x_k^t, z_k^t, u_k^t)\) to converge to a stationary point \((x^*, z^*, u^*)\) of the augmented Lagrangian for problem (13), which is defined by the conditions [Boyd et al. 2011]:

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

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

\[
\hat{g}(z) = g(B^{-1}z), \quad K = AG^{-1}A^T. \tag{26}
\]

We denote by \( \rho(K) \) the spectral radius of matrix \( K \). 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., \( \|\nabla \hat{g}(z_1) - \nabla \hat{g}(z_2)\| \leq L \|z_1 - z_2\| \) \( \forall z_1, z_2 \in \mathbb{R}^n \).

Then we have:

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

\[
\|Bz^{n+1} - Bz^n\| \leq \gamma_1 \|Bz^n - Bz^{n-1}\|,
\]

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

**Theorem 3.2.** If Assumptions 3.1-3.3 are satisfied, \( \rho(K) < \frac{1}{2} \) and \( I - \mu K \) is invertible, then for a sufficiently large \( \mu \) the \( x \)-\( z \)-\( u \) iteration (18)-(20) converges to a stationary point defined in Eq. (25). Moreover,

\[
\|v^{k+1} - v^k\| \leq \gamma_2 \|v^k - v^{k-1}\|,
\]

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

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 \( T(x, z) \) 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 \( \hat{g}(z) \geq 0 \) \( \forall z \in \mathbb{R}^n \). (2) \( \hat{g} \) is continuous on \( \mathbb{R}^n \) and differentiable in \( I_{\text{dom}(\hat{g})} \). (3) \( \hat{g} \) is Lipschitz differentiable on any compact convex set in \( \text{dom}(\hat{g}) \).

For linear convergence of the \( x \)-\( z \)-\( u \) iteration, we assume the following for the initial value \((x^0, z^0, u^0)\) and penalty parameter \( \mu \):

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

\[
c_1 = \sup_{z \in \mathcal{L}_z^\mu} \frac{1}{2\mu} \|B^{-T} \nabla g(z)\|^2
\]

and \( T(x^0, z^0) \). Moreover, suppose \( \text{conv}(\mathcal{L}_z^\mu + c_1) \subset \text{dom}(\hat{g}) \) 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, \( \frac{\mu}{2} - \frac{1}{L_c} > \frac{1}{2L} \), and \( \rho(K) < \frac{1}{L_c} \). Then for a sufficiently large \( \mu \) the \( x \)-\( z \)-\( u \) iteration (15)-(17) converges to a stationary point defined in Eq. (25), and

\[
\|Bz^{n+1} - Bz^n\| \leq \gamma_3 \|Bz^n - Bz^{n-1}\| \quad \text{with} \quad \gamma_3 = \frac{8\rho(K)}{\mu \eta^2} + \frac{4}{\mu} \|B^{-T} \nabla g(z)\|^2,
\]

where \( \eta \) is defined in Proposition 3.5. Moreover, let \( L_d \) be a Lipschitz constant of \( \nabla \hat{g} \) over \( \text{conv}(\mathcal{L}_z^\mu + c_1) \), and suppose \( \text{conv}(\mathcal{L}_z^\mu + c_1) \subset \text{dom}(\hat{g}) \).

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

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; Giselson 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.
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 $\{(y, F(y)) \mid y \in \mathbb{R}^n\} \subset \mathbb{R}^{n+1}$ 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 $\frac{r_k}{z_k} = \frac{r_k}{z_k} > \frac{R_k}{z_k}$. Then for a sufficiently large $\mu$ the $z$-$x$-$u$ iteration (15)-(17) converges to a stationary point defined in Eq. (25), and $\sum_{k=1}^{\infty} \| z_{\mu+1} - z_k \| < \infty$.

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

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

Remark 3.5. Compared with existing convergence results for non-convex ADMM such as [Li and Pong 2015; Wang et al. 2019], for the $z$-$x$-$u$ 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.

### 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]. The source code of our implementation is available at https://github.com/bldeng/AASolver.

![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 cases, 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.](image1)

![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.](image2)
variable range. In the following, for all physical simulation and geometry optimization problems, we 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:

\[
\min_{x,z} f(x) + g(z) \quad \text{s.t.} \quad W(z - Dx) = 0, \tag{28}
\]

Here \( x \) is the node positions of the discretized object, \( f(x) \) is a momentum energy of the form (14) with \( G \) being a scaled mass matrix, \( Dx \) collects the deformation gradient of each element, \( g(z) \) is the elastic potential energy, and \( W \) is a diagonal scaling matrix that improves conditioning. This problem is solved in (28) using ADMM with the \( x,z,u \) iteration. As it satisfies the assumptions in Proposition 3.1, we apply Anderson acceleration to variable \( z \) according to Algorithm 3. Our method is implemented based on the source code released by the authors of [Overby et al. 2017] on https://github.com/mattoverby/admm-elastic. 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 \( z \) and \( u \) together. We can see that accelerating \( z \) 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 \( z \) and the recovered \( u \) 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 \( z \) and \( u \), 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 \( g(z) \), 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, \( g(z) \) 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 \( z \), we cannot apply our method to the \( x,z,u \) iteration (see Remark 3.1). Instead, we adopt the \( x,z,u \) iteration and apply Algorithm 4 to accelerate \( u \) 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
We also apply our method to an ADMM solver for mesh optimization. Therefore, we apply the constraints on the residual for three frames. The input is a mesh with vertex positions \( x \) selected from \( \mathcal{C}_i \) (\( i \in \mathcal{S} \)), and hard constraints \( A_i x \in \mathcal{C}_i \) (\( j \in \mathcal{H} \)). Here each reduction matrix \( A_i \) and \( A_j \) 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 \( z_i \in \mathcal{C}_i \) and solve the problem using ADMM with the augmented Lagrangian method (ALM), where each iteration performs multiple alternate updates of \( z \) and \( x \) followed by a single update of \( u \) 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 \( z-x-u \) 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 \( \left[ \frac{\pi}{4}, \frac{3\pi}{4} \right] \).
- Soft constraint: each vertex lies on a given reference surface.

The mesh is optimized without the Laplacian fairing term, i.e., \( L = 0 \). Our method leads to a faster decrease of the combined residual with 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 \( x \) that prevent the nodes from penetrating the static objects. As this would reduce the \( x \)-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 \( x \) instead and solve the problem using \( z-x-u \) 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 \( x \), soft constraints \( A_i x \in \mathcal{C}_i \) (\( i \in \mathcal{S} \)), and hard constraints \( A_j x \in \mathcal{C}_j \) (\( j \in \mathcal{H} \)). Here each reduction matrix \( A_i \) and \( A_j \) 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 \( z_i \in \mathcal{C}_i \) and solve the problem using ADMM with the augmented Lagrangian method (ALM), where each iteration performs multiple alternate updates of \( z \) and \( x \) followed by a single update of \( u \) 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 \( z-x-u \) 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 \( \left[ \frac{\pi}{4}, \frac{3\pi}{4} \right] \).
- Soft constraint: each vertex lies on a given reference surface.

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

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.

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.
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.

Fig. 8. Comparison of wire mesh optimization 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.

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

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 $\alpha$ and edge length $e$:

$$\xi(e) = \frac{|e - l|}{l}, \quad \gamma(\alpha) = \begin{cases} \frac{\pi}{2} - \alpha & \text{if } \alpha < \frac{\pi}{2}, \\ a - \frac{3\pi}{4} & \text{if } \alpha > \frac{3\pi}{4}, \\ 0 & \text{otherwise.} \end{cases}$$ (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.
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 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 \( d_{\text{max}}(F)/\bar{F} \), where \( d_{\text{max}}(F) \) is the maximum distance from a vertex of \( F \) to the best fitting plane of its vertices, and \( \bar{F} \) 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 \( x \) from an observation image \( f \) with Gaussian noise and a convolution operator \( K \):

\[
\min_{x,z} \lambda_1 \|z_1 - f\|^2 + \lambda_2 \|z_2^{i,j}\| \quad \text{s.t.} \quad Kx = z_1, \ (\nabla x)_{i,j} = z_2^{i,j} \quad \forall i,j.
\]

where \((\nabla x)_{i,j}\) is the image gradient of \( x \) at pixel \((i,j)\). This is solved in [Heide et al. 2016] using ADMM with the \( x-z-u \) 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 \( x \) is computed by solving an optimization problem

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

where \( z \) is an auxiliary variable for image gradient, and \( g(z) \) 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 \( u^k \) can be represented as a function of \( z^k \) via Eq. (49). Therefore, we apply Anderson acceleration to \( z \) alone. Moreover, as \( g(z) \) is quadratic, Eq. (49) implies that \( z^k \) and \( u^k \) are related by a linear map. Thus we use the history \( z \) to compute the affine combination coefficients for Anderson acceleration, and apply them to both \( z \) and \( u \) to derive the accelerated \( z \) and its compatible \( u \), 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
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 count and computational time associated with 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.
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. 61627481), and Youth Innovation Promotion Association CAS (No. 2018495).

REFERENCES


A PROOF FOR PROPOSITION 3.1

By the optimality condition of (16) we have:

\[ \nabla g(z^{k+1}) - \mu B^T(Az^{k+1} - Bz^k + u^k - c) = 0. \]

Put (17) into (33):

\[ \nabla g(z^{k+1}) = \mu B^T u^{k+1}, \]

which completes the proof.

B PROOF FOR PROPOSITION 3.2

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

\[ z^{k+2} = z^{k+1}. \]

Then we have

\[ u^{k+2} = u^{k+1} \]

by (34)

\[ \Rightarrow Ax^{k+2} - Bz^{k+2} - c = 0 \]

by (17)

\[ \Rightarrow Ax^{k+2} - Bz^{k+1} - c = 0 \]

by (35).

For the second part, if \( Ax^{k+2} - Bz^{k+1} - c = 0 \) then from (17):

\[ Ax^{k+2} - c + u^{k+1} = Ax^{k+1} - c + u^{k}. \]
And from (16) and Remark 3.2
\[ z^{k+1} = \arg\min_z \left( g(z) + \frac{\mu}{2} \|Ax^{k+1} - z + u^k\|^2 \right) \]
\[ = \arg\min_z \left( g(z) + \frac{\mu}{2} \|Ax^{k+1} - z - Bz_k\|^2 \right) = z^{k+2}, \]
which completes the proof. □

C PROOF FOR PROPOSITION 3.3
By (20) we have:
\[ Bz^{k+1} - u^k + c = Ax^{k+1} - u^{k+1} \quad \text{(37)} \]

Put (37) into (19):
\[ (G + \mu A^T)Ax^{k+1} = G\tilde{x} + \mu A^T(Ax^{k+1} - u^{k+1}) \]
\[ \Rightarrow Gx^{k+1} = G\tilde{x} - \mu A^T u^{k+1} \]
\[ \Rightarrow x^{k+1} = \tilde{x} - \mu G^{-1} A^T u^{k+1}, \quad \text{(38)} \]
which completes the proof. □

D PROOF FOR PROPOSITION 3.4
For the first part, suppose \( u^{k+1} \) is the fixed-point of the \( z-x-u \) iteration, so that
\[ u^{k+2} = u^{k+1}. \quad \text{(39)} \]
Then by (38) and (39):
\[ x^{k+2} = x^{k+1}. \quad \text{(40)} \]
Therefore
\[ Ax^{k+2} - Bz^{k+2} = 0 \quad \text{by (20) and (39)} \]
\[ \Rightarrow Ax^{k+1} - Bz^{k+2} = 0 \quad \text{by (40)}. \]
For the second part, suppose
\[ Ax^{k+1} - Bz^{k+2} - c = 0. \quad \text{(41)} \]
Then we have
\[ u^{k+1} - Bz^{k+2} = u^k - Bz^{k+1} \quad \text{by (20) and (41)} \]
\[ \Rightarrow x^{k+2} = x^{k+1} \quad \text{by (19)} \]
\[ \Rightarrow Ax^{k+2} - Bz^{k+2} = 0 \quad \text{by (41)} \]
\[ \Rightarrow u^{k+2} = u^{k+1} \quad \text{by (20)}, \]
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:
\[ \min_{x,z} f(x) + g(z) \quad \text{s.t. } Ax - Bz = c. \quad \text{(42)} \]
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 \( x-z-u \) iteration is similar:
\[ x^{k+1} = \arg\min_x L(x, z^k, u^k), \quad \text{(43)} \]
\[ z^{k+1} = \arg\min_z L(x^{k+1}, z, u^k), \quad \text{(44)} \]
\[ u^{k+1} = u^k + Ax^{k+1} - Bz^{k+1} - c. \quad \text{(45)} \]
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:
\[ -A^T(Ax^{k+1} - Bz^k - c + u^k) \in \partial f(x^{k+1}), \quad \text{(46)} \]
\[ B^T(Ax^{k+1} - Bz^{k+1} - c + u^k) \in \partial g(z^{k+1}), \quad \text{(47)} \]
\[ u^{k+1} = u^k + Ax^{k+1} - Bz^{k+1} - c. \quad \text{(48)} \]

E.1 \( u \) as a function of \( z \)
By (47) and (48):
\[ B^T u^{k+1} \in \partial g(z^{k+1}). \quad \text{(49)} \]
Now we can see that \( u^{k+1} \) is a function \( z^{k+1} \) if and only if:

(C2) \( B \) is invertible.

(C3) \( \partial g(z) \) contains exactly one element \( \forall z \in \text{dom}(\partial g) \).

We need additional conditions in order to use Anderson acceleration on \( z \). Note that Anderson acceleration generates \( z^{k+1} \) by affine combination. So if we want to use (49) to compute \( u_{z_{AA}} \) from \( z_{AA} \), the following condition is needed:

(C4) The domain of \( \partial g \), defined as \( \{z \mid \partial g(z) \neq 0\} \), 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 \( \partial g(z) \) is a single-valued mapping.

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

(C5') \( g(z) \) 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 \( \partial g \), defined as \( \bigcup_{z \in R^n} \partial g(z) \), 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 \( c = 0 \) in Eq. (13) to simplify notation.
F.1 Proof for Theorem 3.5

Recall that Theorem 3.5 is about general convergence of the \( x \cdot z \cdot u \) iteration. We first note that:

\[
\nabla \hat{g}(z) = B^{-T} \nabla g(B^{-1}z) \tag{50} \\
\n\implies \nabla \hat{g}(Bz) = B^{-T} \nabla g(z). \tag{51}
\]

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 \( L_k \) in Assumption 3.5 we have:

**Proposition F.1.** Suppose the Lipschitz constant of \( \nabla g(z) \) over \( \text{conv}(\mathcal{D}_a) \) is \( L_1 \), then \( \forall Bz_1, Bz_2 \in \mathcal{D}_a \), we have

\[
|\hat{g}(Bz_1) - \hat{g}(Bz_2)| - (\nabla \hat{g}(Bz_1), Bz_1 - Bz_2) | \leq \frac{L_1}{2} \| Bz_1 - Bz_2 \|^2. \tag{52}
\]

Moreover, if \( \mu > L_1 \) and \( z_2 \in \arg\min \{ g(z) + \frac{\mu}{2} \| Bz - q \|^2 \} \), then:

\[
g(z_2) + \frac{\mu}{2} \| Bz_2 - q \|^2 \leq g(z_1) + \frac{\mu}{2} \| Bz_1 - q \|^2 - \frac{\mu - L_1}{2} \| Bz_1 - Bz_2 \|^2. \tag{53}
\]

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, \( \frac{\mu}{2} - \frac{L_1}{\mu} > \frac{L_1}{2} \), and the \( x \cdot z \cdot u \) iteration satisfies \( g(x^k) \leq T(x^0, z^0) + c_1 \) and \( L(x^k, z^k, u^k) \leq L(x^0, z^0, u^0) = T(x^0, z^0) \). Then

\[
g(x^{k+1}) \leq T(x^0, z^0) + c_1, L(x^{k+1}, z^{k+1}, u^{k+1}) \leq T(x^0, z^0). \tag{54}
\]

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

\[
g(x^{k+1}) + \frac{\mu}{2} \| Ax^{k+1} - Bz^{k+1} + u^k \|^2 \leq g(x^k) + \frac{\mu}{2} \| Ax^k - Bz^k + u^k \|^2.
\]

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

\[
f(x^{k+1}) + \frac{\mu}{2} \| Ax^{k+1} - Bz^k + u^k \|^2 \leq f(x^k) + \frac{\mu}{2} \| Ax^k - Bz^k + u^k \|^2.
\]

Combine the two equations above:

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

By (17) and (34):

\[
T(x^{k+1}, z^{k+1}) + \frac{\mu}{2} \| u^{k+1} \|^2 \leq L(x^k, z^k, u^k) + \frac{1}{2\mu} \| B^{-T} \nabla g(x^k) \|^2. \tag{56}
\]

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

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

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

\[
L(x^{k+1}, z^{k+1}, u^k) \leq L(x^k, z^k, u^k), \tag{57}
\]

\[
L(x^{k+1}, z^{k+1}, u^{k+1}) \leq L(x^{k+1}, z^k, u^k) - \frac{\mu}{2} \| Bz^{k+1} - Bz^k \|^2. \tag{58}
\]

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 \( \frac{\mu}{2} - \frac{L_1}{\mu} > \frac{L_1}{2} \):

\[
L(x^{k+1}, z^{k+1}, u^{k+1}) \leq L(x^k, z^k, u^k) - \frac{\mu}{2} \frac{L_1}{\mu} - \frac{L_1}{2} \| Bz^{k+1} - Bz^k \|^2.
\]

Which completes the proof. \( \square \)

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

**Proposition F.2.** Suppose Assumptions 3.4 and 3.5 hold, and \( \frac{\mu}{2} - \frac{L_1}{\mu} > \frac{L_1}{2} \). Then the \( x \cdot z \cdot u \) iteration satisfies

\[
g(x^k) \leq T(x^0, z^0) + c_1, L(x^k, z^k, u^k) \leq T(x^0, z^0). \tag{59}
\]

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 \( (x^k, z^k, u^k) \):

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

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

\[
L(x^k, z^k, u^k) - L(x^{k+1}, z^{k+1}, u^{k+1}) \geq C_1(\mu)\| B(z^{k+1} - z^k) \|^2 + \| A(x^{k+1} - x^k) \|^2.
\]

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

\[
\| d_k \|^2 \leq C_2(\mu)\| (Bz^{k+1} - z^k) \| + \| A(x^{k+1} - x^k) \|.
\]

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

\[
\lim_{s \to \infty} L(x^k, z^k, u^k) = L(x^s, z^s, u^s).
\]

Note that although the \( x \cdot z \cdot u \) iteration is not same as the one defined in [Li and Pong 2015], the proof for [Li and Pong 2015, Theorem 3.5] 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 3.5], and Proposition F.2 in our paper, we only need to show (P1)-(P4) hold for \( (x^k, z^k, u^k) \).

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

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

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

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

\[
L(x^{k+1}, z^{k+1}, u^{k+1}) \leq L(x^k, z^k, u^k) - \| x^{k+1} - x^k \|^2. \tag{62}
\]
Here \( l > 0 \) is some constant. (62),(57) and (58) show that (P2) holds. (P4) is trivial for our problem. For (P3) we have:

\[
\nabla_x L(x^{k+1}, z^{k+1}, u^{k+1}) = G(x^{k+1} - x) + \mu A^T(Ax^{k+1} - Bz^{k+1} + u^{k+1})
\]

\[
= \mu A^T(Bz^{k} - Bz^{k+1} + u^{k+1} - u^{k}),
\]

(63)

\[
\nabla_z L(x^{k+1}, z^{k+1}, u^{k+1}) = \nabla g(z^{k+1}) + \mu B^T(Bz^{k+1} - Ax^{k+1} - u^{k+1})
\]

\[
= \mu B^T(u^{k+1} - u^{k}),
\]

(64)

\[
\nabla_u L(x^{k+1}, z^{k+1}, u^{k+1}) = \mu (Ax^{k+1} - Bz^{k+1}) = \mu (u^{k+1} - u^{k}).
\]

(65)

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

\[
\|\nabla_x L(x^{k+1}, z^{k+1}, u^{k+1})\| \leq \sqrt{\rho(A^T A)(\mu + L_c)}\|Bz^{k+1} - Bz^k\|,
\]

\[
\|\nabla_z L(x^{k+1}, z^{k+1}, u^{k+1})\| \leq \sqrt{\rho(B^T B)L_c}\|Bz^{k+1} - Bz^k\|.
\]

And notice that \( \nabla_u L(x^{k+1}, z^{k+1}, u^{k+1}) = -B^T \nabla_x L(x^{k+1}, z^{k+1}, u^{k+1}), \) then we get the result. \( \square \)

\[\text{}\]

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:

\[
N(z) := z + \frac{1}{\mu} B^{-T} \nabla g(B^{-1} z).
\]

(66)

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

\[
N(Bz^{k+1}) = (1 + \mu K)^{-1}(Ax + \mu KBz^k + \frac{1}{\mu} B^{-T} \nabla g(z^k)),
\]

(67)

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

**Proof.** By (17) we have:

\[
Bz^k - u^{k+1} = Ax^{k+1} + Bz^k - Bz^{k+1} - u^{k+1}.
\]

(by (15))

\[
(1 + \mu A^{-1} A^T)(u^{k+1} + Bz^{k+1}) = \frac{1}{\mu} B^{-T} \nabla g(z^k).
\]

(by (17))

\[
(1 + \mu A_1 G^{-1} A_1^T)(u^{k+1} + Bz^{k+1}) = \mu G^{-1} A_1^T Bz + \frac{1}{\mu} B^{-T} \nabla g(z^k).
\]

(by (34))

From the definitions of \( N \) and \( K \), the last equation above becomes:

\[
(I + \mu K) N(Bz^{k+1}) = Ax + \mu KBz^k + \frac{1}{\mu} B^{-T} \nabla g(z^k)
\]

\[
N(Bz^{k+1}) = (1 + \mu K)^{-1}(Ax + \mu KBz^k + \frac{1}{\mu} B^{-T} \nabla g(z^k)),
\]

which completes the proof. \( \square \)

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

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

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

\[
x^* = (G + \mu A^T A)^{-1}(Gx + \mu A^T (Bz^* + e - u^*))
\]

(68)

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

\[
Ax^* - Bz^* = e
\]

(69)

The identity \( \nabla f(x^*) + A^T u^* = 0 \) then follows from (68) and (69). \( \square \)

We now show that \( \{z^k\} \) converge linearly:

**Proof for Theorem 3.3.** From (67):

\[
N(Bz^{k+1}) - N(Bz^k) = (I + \mu K)^{-1}(\mu KBz^k - z^{k-1}) + \frac{1}{\mu} B^{-T} \nabla g(z^k) - \nabla g(z^{k-1})
\]

(70)

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

\[
\|\nabla g(Bz^{k+1}) - \nabla g(Bz^k)\| \leq L_c\|Bz^{k+1} - Bz^k\|, \forall k \in \mathbb{N}
\]

\[
\Rightarrow \|N(Bz^{k+1}) - N(Bz^k)\| \leq (1 - \frac{L_c}{\mu})\|Bz^{k+1} - Bz^k\|.
\]

(71)

For the right hand side of (70):

\[
\|Bz^{k+1} - Bz^k\| = \|I + \mu K\|^{-1}(\mu KBz^k - z^{k-1}) + \frac{1}{\mu} B^{-T} \nabla g(z^k) - \nabla g(z^{k-1})
\]

(72)

And notice that \( K \) is positive semi-definite:

\[
\|\frac{1}{\mu} (I + \mu K)^{-1} B^{-T} \nabla g(z^k) - \nabla g(z^{k-1})\| \leq \frac{L_c}{\mu}\|Bz^k - Bz^{k-1}\|.
\]

(73)

Combine (72) with (73):

\[
\|I + \mu K\|^{-1}(\mu KBz^k - Bz^{k-1}) + \frac{1}{\mu} B^{-T} \nabla g(z^k) - \nabla g(z^{k-1})\|
\]

\[
\leq \frac{\mu \rho(K)}{1 + \mu \rho(K)} \|Bz^k - Bz^{k-1}\|.
\]

(74)

By (71) and (74) we have:

\[
\|Bz^{k+1} - Bz^k\| \leq \frac{\mu \rho(K)}{1 + \mu \rho(K)} \frac{L_c}{\mu}\|Bz^k - Bz^{k-1}\|.
\]

(75)

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

**Proof for Proposition 3.5.** By the definition of $K$ in (26), we know that $K(R(A)) \subset R(A)$ since $R(A)$ is a linear subspace and $K$ is a linear operator, for the proof it suffices to show $\ker(K) \cap R(A) = \{0\}$, where $\ker(K)$ is the kernel of $K$. Now assume $y \in \ker(K)$, then for any $z \in \mathbb{R}^9$ where $q$ is the number of rows in matrix $A$, we have:

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

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

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

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

$$u^{k+1} = Ax^{k+1} - Ax^k + \frac{1}{\mu}B^{-T}\nabla g(x^{k+1}).$$

**Proof.** From (20):

$$u^{k+1} - Ax^{k+1} = u^k - Bz^{k+1}.$$  

From (23):

$$Ax^k + u^k = Ax - \mu Ku^k + u^k$$

by (18)

$$Bz^{k+1} + \frac{1}{\mu}B^{-T}\nabla g(z^{k+1}) = Ax^k + u^k$$

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

Combine (77) and (78) then we can get the result. $\square$

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

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

$$\|u^k\|^2 \leq \frac{4}{\eta^2 \mu^2} \|Ax^k - Ax^k - \nabla g(\mu z^{k+1}) \|^2 + \frac{2}{\mu} \|B^{-T}\nabla g(z^{k+1})\|^2$$

$$\|u^{k+1} - u^k\|^2 \leq \frac{4}{\mu^2 \eta^2} \|Ax^{k+1} - Ax^k\|^2 + \frac{2}{\mu^2} \|B^{-T}\nabla g(z^{k+1})\|^2.$$  

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

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

And from (23):

$$Ax^{k+1} = Ax - \mu Ku^{k+1}$$

by (76)

$$Ax^{k+1} = Ax - \mu (Ax^{k+1} - Ax^k) - KB^{-T}\nabla g(z^{k+1})$$

$$\implies Ax^{k+1} - Ax + KB^{-T}\nabla g(z^{k+1}) = -\mu K(Ax^{k+1} - Ax^k).$$

Hence by Proposition 3.5:

$$\mu^2 \eta^2 \|Ax^{k+1} - Ax^k\|^2 = 2\|Ax^{k+1} - Ax^k\|^2 + 2p(K)^2 \frac{1}{\mu} \|B^{-T}\nabla g(z^{k+1})\|^2.$$  

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

$$u^{k+1} - u^k = A(x^{k+1} - 2x^k + x^{k-1}) + \frac{1}{\mu}B^{-T}(\nabla g(x^{k+1}) - \nabla g(x^k))$$

$$\implies \|u^{k+1} - u^k\|^2 \leq 2\|A(x^{k+1} - 2x^k + x^{k-1})\|^2$$

$$\quad + \frac{2}{\mu^2} \|B^{-T}(\nabla g(x^{k+1}) - \nabla g(x^k))\|^2.$$  

And by (83):

$$Ax^{k+1} - Ax^k + KB^{-T}(\nabla g(x^{k+1}) - \nabla g(x^k)) = -\mu KA(x^{k+1} - 2x^k + x^{k-1}).$$

Hence:

$$\mu^2 \eta^2 \|A(x^{k+1} - 2x^k + x^{k-1})\|^2 \leq 2\|Ax^{k+1} - Ax^k\|^2 + 2p(K)^2 \|B^{-T}(\nabla g(x^{k+1}) - \nabla g(x^k))\|^2.$$  

Then (80) follows from (84) and (85). $\square$

Similar to Proposition F.2, we can prove:

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

$$T(x^k, z^k) \leq T(x^0, z^0) + c_2 + c_3, L(x^k, z^k, u^k) \leq T(x^0, z^0) + c_3.$$  

**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 $x^{l+1}$ in (18):

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

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

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

add (88) to (87):

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

By induction:

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

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

$$\frac{\mu}{2} \|u^l\|^2 \leq \frac{2}{\eta^2 \mu} \|Ax^l - Ax^l\|^2 + \frac{2p(K)^2}{\mu^2} \|B^{-T}\nabla g(\mu z^{l+1})\|^2.$$  

By induction:

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

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

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

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:

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

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

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

By Assumption 3.2:

$$L(x^0, z^0, u^0) \leq L(x^0, z^0, u^0) - \frac{\mu}{2} \|Ax^1 - Ax^0\|^2.$$
Moreover, we have:
\[ L(x^1, z^1, u^1) = L(x^1, z^1, u^0) + \mu \| u^1 - u^0 \|^2 \]
By (79) and \( u^0 = 0 \):
\[ \mu \| u^1 - u^0 \|^2 = \mu \| u^1 \|^2 \]
\[ = \frac{4}{\eta^2 \mu} \| Ax^1 - Ax \|^2 + \left( \frac{4 \rho(K)^2}{\eta^2 \mu^2} + \frac{2}{\mu} \right) \| B^T \nabla g(z^1) \|^2. \]

Moreover, we have:
\[ \| B^T \nabla g(z^1) \|^2 \leq 2 \| B^T \nabla g(z^1) - B^T \nabla g(z^0) \|^2 + 2 \| B^T \nabla g(z^0) \|^2 \]
\[ \leq 2L_d \| Bz^1 - Bz^0 \|^2 + 2 \| B^T \nabla g(z^0) \|^2. \]
So if \( \frac{\mu}{2} \geq \frac{4}{\eta^2 \mu} \) and \( \mu \frac{L_d}{\mu^2} \geq \frac{4}{\eta^2 \mu} \), then we have:
\[ L(x^1, z^1, u^1) \leq L(x^0, z^0, u^0) + \left( \frac{4 \rho(K)^2}{\eta^2 \mu^2} + \frac{4}{\mu} \right) \| B^T \nabla g(z^0) \|^2 \]
\[ = (T(x^0, z^0) + \frac{8 \rho(K)^2}{\eta^2 \mu^2} + \frac{4}{\mu} \| B^T \nabla g(z^0) \|^2. \]

By the definition of \( c \) we have \( L(x^1, z^1, u^1) \leq T(x^0, z^0) + c_3 \). Now suppose \( l = 1 \). Similar to the proof of the case \( l = 0 \) we have:
\[ L(x^1, z^{l+1}, u^l) \leq L(x^1, z^1, u^1) - \frac{\mu}{2} \| Bz^{l+1} - Bz^1 \|^2, \]
\[ L(x^{l+1}, z^{l+1}, u^l) \leq L(x^1, z^{l+1}, u^1) - \frac{\mu}{2} \| Ax^{l+1} - Ax^1 \|^2, \]
\[ L(x^{l+1}, z^{l+1}, u^{l+1}) = L(x^{l+1}, z^{l+1}, u^1) + \mu \| u^{l+1} - u^1 \|^2. \]

By (80) we have:
\[ \mu \| u^{l+1} - u^1 \|^2 \leq \frac{4}{\eta^2 \mu^2} \| Ax^{l+1} - Ax^1 \|^2 \]
\[ + \left( \frac{4 \rho(K)^2}{\eta^2 \mu^2} + \frac{2}{\mu} \right) \| B^T \nabla g(z^{l+1}) - \nabla g(z^1) \|^2. \]

Since \( g(z^1), g(z^{l+1}) \leq T(x^0, z^0) + c_2 + c_3 \), by the definition of \( L_d \):
\[ \| B^T (\nabla g(z^{l+1}) - \nabla g(z^1)) \| \leq L_d \| Bz^{l+1} - Bz^1 \|. \quad (89) \]

Hence we have:
\[ \mu \| u^{l+1} - u^1 \|^2 \leq \frac{4}{\eta^2 \mu^2} \| Ax^{l+1} - Ax^1 \|^2 \]
\[ + \left( \frac{4 \rho(K)^2 L_d^2}{\eta^2 \mu^2} + \frac{2 L_d^2}{\mu} \right) \| Bz^{l+1} - Bz^1 \|^2. \]

If \( \frac{\mu}{2} \geq \frac{4}{\eta^2 \mu} \) and \( \mu \frac{L_d}{\mu^2} \geq \frac{4}{\eta^2 \mu} \), then we have:
\[ L(x^{l+1}, z^{l+1}, u^{l+1}) \leq L(x^1, z^1, u^1) \leq T(x^0, z^0) + c_3 \quad (90) \]
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 \( \{u^k\} \) converges. Then for the main proof:

Proof for Theorem 3.4. By (78):
\[ Bkz_k + \frac{1}{\mu} B_z g(z_k) = Ax - v_k \]
\[ \Rightarrow B(z_{k+1} - z_k) + \frac{1}{\mu} B_z (g(z_{k+1}) - g(z_k)) = (v_k - v_{k-1}). \]

By (89):
\[ (1 - \frac{L_d}{\mu}) \| Bz_{k+1} - Bz_k \| \leq \| v_k - v_{k-1} \|. \]

Hence we have:
\[ \frac{1}{\mu} \| B_z g(z_{k+1}) - g(z_k) \| \leq \frac{L_d}{\mu} \| Bz_{k+1} - Bz_k \| \]
\[ \leq \frac{L_d}{\mu - L_d} \| v_k - v_{k-1} \|. \quad (91) \]

By (76) and (82):
\[ (I + \mu K)u_k = \mu Ku_k + \frac{1}{\mu} B_z g(z_{k+1}) \]
\[ \Rightarrow v_{k+1} = (I + \mu K)^{-1} \mu Ku_k + (I + \mu K)^{-1}(I - \mu K)^{-1} B_z g(z_{k+1}). \]

Hence we have:
\[ \| v_{k+1} - v_k \| \leq \| (I + \mu K)^{-1} \mu Ku_k - v_{k-1} \| \]
\[ + \frac{1}{\mu} \| (I + \mu K)^{-1}(I - \mu K)B_z g(z_{k+1}) - g(z_k) \|. \]

Similar to (72) we have:
\[ \| (I + \mu K)^{-1} \mu K(v_k - v_{k-1}) \| \leq \frac{\rho(K)}{1 + \rho(K)} \| v_k - v_{k-1} \| \]
\[ \Rightarrow \| v_{k+1} - v_k \| \leq \left( \frac{\rho(K)}{1 + \rho(K)} + \frac{L_d}{\mu - L_d} \right) \| v_k - v_{k-1} \|. \]

Then let \( \mu \geq \max \left\{ -\frac{2}{\eta^2 - \rho(K)}, \frac{1}{L_d} \right\} \) 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.