

BC-ADMM: A Parallel Decoupled Non-convex Constrained Optimizer for Robot Applications

Journal Title
XX(X):1–28
©The Author(s) 2016
Reprints and permission:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/ToBeAssigned
www.sagepub.com/

SAGE

Zherong Pan and Kui Wu¹

Abstract

Non-convex constrained optimizations are ubiquitous in robotic applications such as multi-agent navigation, UAV trajectory optimization, and soft robot simulation. As a common feature in these problems, the associated non-convex constraints, including collision constraints, inversion-free constraints, and strain limits, are also non-smooth with ill-defined gradients. It is well-known that such constraints are notoriously difficult to handle, for which off-the-shelf optimizers can fail catastrophically. Instead, prior works tend to design problem-specific optimizers that trade performance for robustness. To efficiently solve this problem class in a unified manner, we propose a variant of Alternating Direction Method of Multiplier (ADMM), called BC-ADMM. Over the past decade, ADMM has achieved great success in efficiently solving many large-scale (constrained) optimization problems by decoupling them into sub-problems that can be solved in parallel. However, prior ADMM algorithms do not have convergence guarantee when dealing with a large number of non-convex constraints with loopy constraint graphs. Instead, our BC-ADMM relaxes each non-convex constraint into a bi-convex function, further breaking the constraint into two sub-problems. We show that such relaxation leads to both theoretical convergence speed guarantees and practical convergence guarantees in the asymptotic sense. Through numerical experiments in a row of four robotic applications, we show that BC-ADMM has faster convergence than conventional gradient descent and Newton's method in terms of wall clock time.

Keywords

Constrained Optimization, Multi-agent Navigation, UAV Trajectory Optimization, Robot Simulation

Introduction

Non-convex constrained optimization problems are pervasive in robotics, particularly in areas such as geometric processing, motion planning, soft-robot modeling, and multi-agent navigation. These challenges commonly arise in applications like deformable body simulation Lipson (2014); Kovalsky et al. (2015), collision-free trajectory optimization Zhang et al. (2020), and multi-robot coordination Guy et al. (2009), among others. Solving such problems is notoriously difficult for two main reasons. First, they often involve large-scale formulations with a high number of decision variables and constraints. For instance, the number of variables can grow superlinearly with mesh resolution. In multi-agent navigation, the number of collision constraints increases quadratically with the number of agents-reaching on the order of 10^4 constraints for 10^2 agents in our evaluations. Similarly, in soft robot modeling using the finite element method, the number of elements can reach 10^3 , while collision and strain-limiting constraints may scale up to 10^6 . Second, many of these non-convex constraints are also non-smooth, often lacking well-defined gradients. A common example is the collision avoidance constraint. To illustrate this challenge, consider the simple scenario shown in Figure 1, where a circular robot attempts to reach a target position while avoiding a triangle-shaped obstacle. This

problem can be formulated as the following optimization:

$$\begin{aligned} \underset{x_1}{\operatorname{argmin}} \quad & \|x_1 - x_1^*\|^2 \\ \text{s.t.} \quad & \inf_{x' \in \text{CH}(x_2, x_3, x_4)} \|x_1 - x'\| \geq r, \end{aligned} \quad (1)$$

which involves a single non-convex collision constraint. It is well-known that this constraint function is non-smooth due to the point-to-set distance function $\inf_{x' \in \text{CH}(x_2, x_3, x_4)} \|x_1 - x'\|$. Indeed, as illustrated in Figure 2, such a distance function has a vanishing gradient when $x_1 \in \text{CH}^\circ(x_2, x_3, x_4)$ where CH° indicates the interior of the convex hull, but a unit-norm directional derivative exists when $x_1 \notin \text{CH}^\circ(x_2, x_3, x_4)$.

Solving constrained optimizations with non-smooth, non-convex constraint functions is notoriously difficult for general-purpose optimization algorithms, such as sequential quadratic programming and primal-dual interior point method Nocedal and Wright (1999), as implemented in mature software packages SNOPT Gill et al. (2005) and IPOPT Wachter (2002). All these algorithms are infeasible point optimizers, meaning that they do not assume the intermediary solutions during an optimization are strictly feasible. Instead, they rely on the gradient of constraints to be non-vanishing, which could be used to recover feasibility from infeasible initial guesses. Unfortunately, such non-vanishing-gradient assumptions on the constraint functions

¹LIGHTSPEED

Email: {zherong.pan.usa@gmail.com, walker.kui.wu@gmail.com}

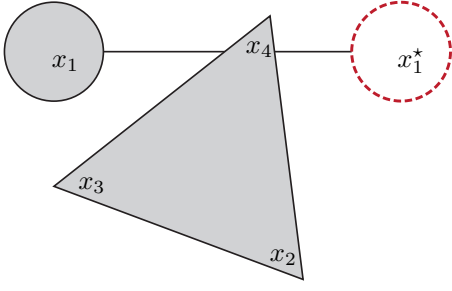


Figure 1. We optimize the position of a circular robot with a radius r under collision constraint, which is located at $x_1 \in \mathbb{R}^2$. The robot needs to reach the goal position x_1^* , while avoiding the triangular obstacle spanned by its three vertices x_2, x_3, x_4 . We can formulate this problem as an optimization with the collision constraint $\inf_{x' \in \text{CH}(x_2, x_3, x_4)} \|x_1 - x'\| \geq r$. Here CH denotes the closed convex set spanned by a set of vertices.

are frequently violated in practical problems, e.g. when $x_1 \in \text{CH}^\circ(x_2, x_3, x_4)$ in Equation 1. The violation of these assumptions can lead to catastrophic failure of infeasible point optimizers by getting stuck at an infeasible solution, as analyzed in Schulman et al. (2014); Byravan et al. (2014).

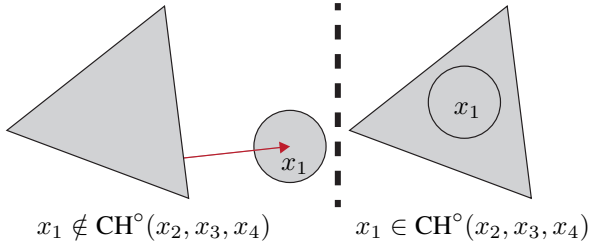


Figure 2. We illustrate two cases of the non-smooth collision constraint function. Left: When $x_1 \notin \text{CH}^\circ(x_2, x_3, x_4)$, the function is locally differentiable with a gradient (red arrow) satisfying $\|\partial[\inf_{x' \in \text{CH}(x_2, x_3, x_4)} \|x_1 - x'\| - r] / \partial x_1\| = 1$. Right: When $x_1 \in \text{CH}^\circ(x_2, x_3, x_4)$, then gradient vanishes.

To address the difficulty of infeasible point optimizers, researchers have noticed that many non-smooth constraints are indeed sufficiently smooth in the feasible domain. For example, the point-to-set distance function in Equation 1 is differentiable with unit-norm gradient when restricted to its feasible domain. Therefore, several works have proposed to use a feasible interior point method by using a barrier function to restrict the solution in its feasible domain. In the example of Equation 1, the constrained optimization is transformed into the following unconstrained optimization, which can then be solved using gradient descend or Newton’s method:

$$\underset{x}{\operatorname{argmin}} \|x - x^*\|^2 - \log_\epsilon [\inf_{y \in \text{CH}(y_1, y_2, y_3)} \|x - y\| - r] \quad (2)$$

with $\log_\epsilon$ being a locally supported variant of log-barrier function depending on a hyperparameter ϵ . Starting from a feasible initial guess, the feasible interior point method maintains constraint feasibility and thus avoids the issue of non-smoothness. It can be shown that the function $\log_\epsilon$ can be designed such that the local solution of Equation 2 is also a local solution of Equation 1 as ϵ tends to zero. Variants of such feasible interior point methods have

been proposed to solve problems in deformable body simulation Li et al. (2020); Du et al. (2024), collision-free trajectory optimization Ni et al. (2022); Zhang et al. (2024), and multi-robot navigation Van den Berg et al. (2008); Karamouzas et al. (2017). These works use problem-specific barrier functions to achieve the ideal balance between computational efficacy, robustness, and solution quality. However, despite their robustness, the computational efficacy has been compromised as compared with the infeasible point methods Gill et al. (2005); Wachter (2002). This is because the optimizer needs to take very small step sizes so that none of the constraints are violated. Typically, the global step size is upper bounded by the single, most stringent constraint.

In this work, we propose an unified structure-aware optimizer to improve the computational efficacy of feasible interior point method. Unlike black-box numerical algorithms, structure-aware optimizers Chang et al. (2014); Jiang et al. (2019); Khatana and Salapaka (2022) exploit the properties in objective functions and accelerate computation. Recently, Alternating Direction Method of Multipliers (ADMM), a prominent structure-aware optimization algorithm, has been extensively applied to general physics simulation Overby et al. (2017), geometric processing Ouyang et al. (2020), motion planning Ni et al. (2022) and multi-agent navigation Saravanos et al. (2021) problems. These methods utilize the fact that objective functions and constraints can be decomposed into terms related to small subsets of variables, leading to a set of smaller sub-problems that can be solved efficiently and in parallel. The solutions of these sub-problems are then summarized into a unified set of variables, forming a consensus Boyd et al. (2011). Unlike the infeasible interior point method, where the entire solution is updated using a global step size, ADMM solves each sub-problem independently, allowing them to take much larger step sizes. Further, by allowing these sub-problems to be solved in parallel, ADMM enjoys a low iterative cost, allowing it to quickly converge to an approximate solution for large problem instances.

However, we observe that when the sub-problems imply non-convex constraints and the connectivity between constraints has loops, then existing structured optimization algorithms are not guaranteed to converge. Unfortunately, such problems are ubiquitous in all the aforementioned paradigms. An illustrative toy example is provided in Figure 3 where we optimize collision-free poses for 3 robots. Variants of this problem have been considered in prior works Van den Berg et al. (2008); Karamouzas et al. (2017). In this case, collision constraints exist between each pair of the 3 robots. If we consider each robot as the node of a graph, and each collision constraint is an edge connecting the pair of robots potentially in collision, then the constraint graph has a loop. Surprisingly, existing convergence analyses of the ADMM algorithm can fail for loopy constraint graphs. This limitation prevents ADMM from being adopted in a wide range of robotic applications.

In this work, we propose a variant of ADMM called Biconvex ADMM (BC-ADMM). For a special kind of non-convex optimization that endows a bi-convex relaxation, our algorithm takes as input a strictly feasible solution and maintains its feasibility during the optimization. BC-ADMM adopts the same algorithm pipeline as standard

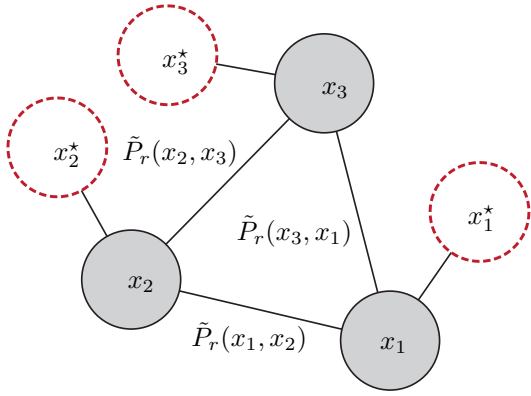


Figure 3. We optimize the position of three circular robots with a uniform radius r under collision constraints, where their position variables are denoted as $x_{1,2,3} \in \mathbb{R}^2$ concatenated into a decision variable $x \in \mathbb{R}^6$. The objective function $f: \mathbb{R}^6 \rightarrow \mathbb{R}^+$ is designed for the three robots to reach their distinctive goals x_i^* , which is defined as $\sum_{i=1}^3 \|x_i - x_i^*\|^2$. Each collision constraint is modeled as an extended-real log-barrier function $\tilde{P} \in \mathbb{R}^4 \rightarrow \mathbb{R}^+ \cup \{\infty\}$ defined as $\tilde{P}_r(x_i, x_j) = -\log_\epsilon(\|x_i - x_j\| - 2r)$, with $\log_\epsilon$ being a locally supported variant of log-barrier function. Altogether, we need three barrier functions $\tilde{P}_r(x_1, x_2)$, $\tilde{P}_r(x_2, x_3)$, and $\tilde{P}_r(x_3, x_1)$, of which the corresponding constraint graph has a loop. Unfortunately, existing ADMM algorithm does not have convergence guarantee for this problem.

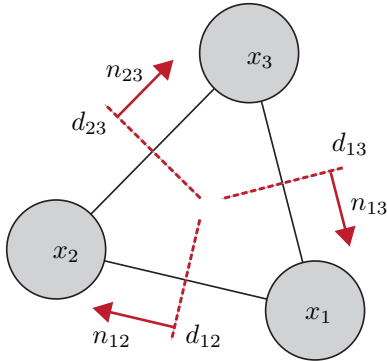


Figure 4. We can reformulate the log-barrier function $\tilde{P}_r(x_i, x_j)$ in Figure 3 into an augmented function $P_r(x_i, x_j, n_{ij}, d_{ij})$. This is done by introducing separating planes between the pair of robots x_i and x_j and require that the two robots lie on different sides of the separating plane, with the normals and offsets of these separating planes (red) denoted as n_{ij} and d_{ij} , respectively. We concatenate all these variables into $z = (n_{12}^T, d_{12}^T, n_{23}^T, d_{23}^T, n_{31}^T, d_{31}^T)^T$. By proper design, we can make the function P_r convex in (x_i, x_j) and in (n_{ij}, d_{ij}) but not both. We refer readers to Section for more details on the design of the function P_r .

distributed ADMM Boyd et al. (2011). Under proper parameter choices, we show that our algorithm has an oracle complexity of $O(\epsilon^{-2})$. Further, by maintaining the feasibility of solutions, BC-ADMM pertains the desirable property similar to anytime motion planner Karaman et al. (2011), aka. it allows the algorithm to be terminated anytime while returning a feasible solution. We propose practical implementations of our method and show that a large class of known optimization problems in motion planning, robot

simulation, and trajectory optimization problems can be modeled after our framework. In the case of Figure 3, we can relax the collision constraint using the celebrated hyperplane separation theorem by inserting a separating plane between each pair of potentially colliding robots as illustrated in Figure 4. Our **main contributions** are summarized below:

- A bi-convex relaxation of a class of non-smooth optimization problems, including **motion planning, soft-robot simulation, multi-agent navigation, and trajectory optimization**.
- A provably robust variant of ADMM for solving the relaxed constrained optimization.

Related Work

In this section, we first review the development and limitations of general-purpose constraint optimizers. Next, we review related works on structured optimization techniques. Finally, we discuss several applications in the robotic community, which involve optimization problems that can be handled by our BC-ADMM.

General-purpose Constrained Optimizations

General-purpose constrained optimization solvers are designed to be applicable for a large problem class with no assumptions on the underlying structures of objective and constraint functions. Typically, they only require these functions to be (twice-)differentiable. Algorithms for constrained optimization include the augmented Lagrangian method, the sequential quadratic programming, and the interior point method. These algorithms have been built into mature software packages, such as LANCELOT Conn et al. (2013), SNOPT Gill et al. (2005), and IPOPT Wachter (2002), and are widely used in optimization problems from robotic applications. However, their convergence to feasible solutions relies on strong assumptions on the Jacobian of constraint functions, which are known as constraint qualifications Boggs and Tolle (1995); Birgin and Martinez (2008). These constraint qualifications ensure that an infeasible solution can be guided back to the feasible domain by the gradients of the constraint functions. Unfortunately, it is non-trivial to check and ensure that constraint qualifications are satisfied for practical problems. Even worse, in problems as illustrated in Figure 1, constraint qualifications are known to fail in many cases. All these factors lead to the lack of robustness in using general-purpose constrained optimizers with failure cases observed and analyzed in several prior works Schulman et al. (2014); Byravan et al. (2014); Pan et al. (2022).

Recently, it has been discovered that several non-smooth constraint functions are indeed smooth when restricted to their feasible domain, which includes collision constraints Li et al. (2020); Karamouzas et al. (2017); Ni et al. (2021), inversion-free constraints Fu and Liu (2016), and strain limits Li et al. (2020). Researchers thus propose to use a feasible interior point method to restrict the solution to its feasible domain, yielding much better robustness than their infeasible counterpart. However, such robustness comes at the cost of computational efficacy Nocedal and

Wright (1999). This is because the feasible interior point method needs to take very small step sizes to ensure all the constraints are satisfied after moving the solution. As a result, the step sizes are typically upper-bounded by the most-stringent constraint.

Structured Optimization Algorithms

Compared with general-purpose algorithms, structured optimizers exploit additional structures in the objective and constraint functions to significantly boost the performance. For example, when both objective and constraints are convex, then globally optimal solutions can be found via local optimization Boyd and Vandenberghe (2004). When the objective function is multi-convex Xu and Yin (2013), then block coordinate descent algorithms can be used with guaranteed convergence. When the objective function is a submodular quadratic function, then binary optimization can be solved efficiently using graph-cut. In our work, we propose a variant of ADMM, which is yet another class of structured optimization algorithms. ADMM works by breaking a large-scale problem into many small problems that can be solved in a numerically stable and efficient manner. Early works on ADMM focus on convex optimization problems Boyd et al. (2011); Deng and Yin (2016). Over the years, the convergence of ADMM has been confirmed for a much wider class of constrained optimization problems with non-convex objectives and possibly non-convex constraints (see e.g. Wang et al. (2019)). The massively parallel nature of ADMM makes it a stellar fit for efficiently solving large-scale problem instances. Indeed, ADMM has been adopted to handle collision and joint constraints for both physics simulation Overby et al. (2017) and geometric processing Zhang et al. (2019). However, we notice that state-of-the-art ADMM convergence analysis requires the associated constraint graph to contain no loops, which is the focus of this work.

Applications of BC-ADMM

We discuss several robotic applications of our BC-ADMM, and we show that all these methods use the feasible interior point method, which suffers from small step sizes.

Multi-agent Navigation In early works, navigating agent motions are simulated using force-based models Reynolds (1987) or continuum models Hughes (2002). However, these methods suffer from numerical stability issues and require careful parameter tuning to generate smooth and collision-free agent motions. Velocity-based methods Van den Berg et al. (2008) are then proposed to introduce hard velocity constraints between a pair of agents to resolve collisions. But these constraints are solved for each agent separately, which limits the allowed range of collision-free motions. Most recently, Karamouzas et al. (2017) draws connections with optimization-based numerical integrators and introduces a formulation that jointly optimizes agent positions at the next timestep under pairwise barrier functions to prevent collisions between agents.

UAV Trajectory Optimization Generating collision-free, executable, and optimal UAV trajectories has been an active area of robotic research Deits and Tedrake (2015); Zhou et al. (2019); Gao et al. (2018); Ni et al. (2021, 2022). Among

other technical disparities, these formulations vary in their techniques for handling the collision constraints. The non-convex collision constraints can be relaxed into a disjoint union of convex constraints Deits and Tedrake (2015); Gao et al. (2018) by pre-computing a set of convex corridor shapes, under which the globally optimal trajectory can be found by solving a mixed-integer convex program. On the other hand, Gao et al. (2018) proposed to introduce non-convex soft penalty functions to search for locally optimal collision-free trajectories, without utilizing safe corridors. Ni et al. (2021) further enhanced the soft-penalty function into a log-barrier function, providing rigorous collision-free guarantees. Unsurprisingly, their underlying feasible interior point optimizer suffers from the small step size. In view of this drawback, follow-up works Ni et al. (2022); Wang et al. (2020) propose to decompose the optimization problem into sub-problems and adopt alternating minimization. However, none of these algorithms can decouple the collision constraints into sub-problems, which significantly limits the potential of alternating minimization.

Deformable Object & Soft Robot Simulation Deformable objects are ubiquitous in robotic manipulation tasks. The simulation, manipulation, and control of deformable bodies have been an active area of research for decades Rus and Tolley (2015). In particular, various formulations have been proposed to model the dynamics of deformable objects, including mass-spring systems Huang et al. (2020); Baraff and Witkin (1998), As-Rigid-As-Possible (ARAP) deformations Sorkine and Alexa (2007); Kwok and Chen (2017); Fang et al. (2018), and the finite element method Duriez (2013); Qin et al. (2024). Whichever formulation is adopted, a practical deformable body simulator needs to handle various non-convex constraints, of which the two most common constraints stem from collision and strain limits Goldenthal et al. (2007); Wang et al. (2010); Kim et al. (2012). The strain limits are introduced to model the behavior where a soft body hardens infinitely under large forces to prevent compression or stretching beyond the material limit. These hard constraints can prevent the underlying optimizer from taking large step sizes and making faster progress. This issue has been partially addressed by prior works Liu et al. (2013); Overby et al. (2017). Unfortunately, these formulations either cannot handle both types of constraints or do not have convergence guarantees.

Problem Formulation

We begin the construction of BC-ADMM by considering the unconstrained optimization problem induced from the feasible interior point method. Considering the following optimization corresponding to the toy problem in Figure 3:

$$\argmin_x \sum_{i=1}^3 \|x_i - x_i^*\|^2 + \tilde{P}_r(x_1, x_2) + \tilde{P}_r(x_2, x_3) + \tilde{P}_r(x_3, x_1), \quad (3)$$

the three barrier potential terms each model a collision constraint between a pair of robots. However, if solve Equation 3 using a global step size, then the step size must be upper bounded by the most stringent constraint, which often-times hinders convergence speed. Instead, ADMM works by

decoupling the constraint barrier functions from the objective function. Specifically, we define the objective function as $f(x) = \sum_{i=1}^3 \|x_i - x_i^*\|^2$ and the barrier functions are summarized into another function:

$$\tilde{g}(y) = \tilde{P}_r(y_1, y_2) + \tilde{P}_r(y_3, y_4) + \tilde{P}_r(y_5, y_6), \quad (4)$$

with $y \in \mathbb{R}^{12}$. ADMM then works by reformulating Equation 3 into the following equivalent form:

$$\operatorname{argmin}_x f(x) + \tilde{g}(Ax), \quad (5)$$

which is our main problem under consideration with a general matrix A . In our toy problem, the matrix is defined as:

$$A = \begin{pmatrix} I & 0 & 0 & 0 & I & 0 \\ 0 & I & I & 0 & 0 & 0 \\ 0 & 0 & 0 & I & 0 & I \end{pmatrix}^T \in \mathbb{R}^{12 \times 6}. \quad (6)$$

We further take the following assumption for Equation 5:

Assumption 1. i) $f: \mathbb{R}^n \rightarrow \mathbb{R}^+ \cup \{\infty\}$ is continuous and differentiable in $\operatorname{dom}(f)$ and $\tilde{g}: \mathbb{R}^m \rightarrow \mathbb{R}^+ \cup \{\infty\}$ is continuous and twice continuously differentiable in $\operatorname{dom}(\tilde{g})$. ii) $f(\bullet) + \tilde{g}(A\bullet)$ is coercive. iii) A strictly feasible $x^0 \in \mathbb{R}^n$ is given, such that $f(x^0) + \tilde{g}(Ax^0) < \infty$.

Many underlying problems in geometric processing, physics simulation, and trajectory generation satisfy our assumptions. For example, in inversion-free deformation, f is a quadratic objective function, $\tilde{g}$ is the elastic energy with log-barrier functions to limit the strain, and the rest shape of the deformable body is strictly feasible and known. For multi-agent trajectory generation, f is the trajectory smoothness objective, $\tilde{g}$ is the collision-free constraint, and the initial state of agents is collision-free and known.

Remark 2. The two properties of function $\tilde{g}$ are both critical to the convergence analysis of our algorithm. As an example, the function that is frequently considered in our applications is the log-barrier function $\tilde{g}(x) = -\log(x)$ (or its locally supported variant $\tilde{g}(x) = -\log_\epsilon(x)$ with similar properties), as in illustrative problem of Figure 3. This function has $\operatorname{dom}(\tilde{g}) = \{x|x > 0\}$ in which it is smooth and thus twice continuously differentiable. In addition, it is continuous in the entire domain $\mathbb{R}$. As a counter-example, we consider the function $\tilde{g}(x) = \iota_{\{x|x>0\}}(x)$, where $\iota_{\mathcal{Z}}(z)$ is the indicator function, which takes value 0 when $z \in \mathcal{Z}$ and ∞ otherwise. We again have $\operatorname{dom}(\tilde{g}) = \{x|x > 0\}$ in which it is constant and thus twice continuously differentiable. However, this function is not continuous at $x = 0$.

ADMM Relaxation

To adopt structured optimization, ADMM reformulates Equation 5 using a slack variable y into the following equivalent form:

$$\operatorname{argmin}_{x,y} f(x) + \tilde{g}(y) \quad \text{s.t. } Ax = y. \quad (7)$$

The above reformulation allows ADMM to introduce the following augmented Lagrangian function associated

with Equation 7:

$$\mathcal{L}(x, y, \lambda) = f(x) + \tilde{g}(y) + \frac{\beta}{2} \|Ax - y\|^2 + \lambda^T (Ax - y). \quad (8)$$

It is well-known that any solution of Equation 7 corresponds to the critical point of the Lagrangian function in all the parameters x , y , and λ . The Lagrangian function is widely used to gauge convergence in the augmented Lagrangian method Conn et al. (2013) and sequential quadratic programming Boggs and Tolle (1995), and we refer readers to these works for more discussions. As a key difference from these methods, however, ADMM searches for the critical point in a decoupled manner by updating parameters x , y , and λ sequentially. The main benefit of this strategy is that each subproblem has a small size and can be solved extremely efficiently. Taking the toy function $\tilde{g}$ from Equation 4 for example, the y update can be decomposed into three subproblems related to (y_1, y_2) , (y_3, y_4) , and (y_5, y_6) , respectively, which can be solved in parallel.

For solving Equation 7, several variants of ADMM algorithms Deng and Yin (2016); Jiang et al. (2019); Themelis and Patrinos (2020) have been developed. However, we show that these algorithms all assume additional properties on the problem data that do not hold in the aforementioned problem instances. In particular, Deng and Yin (2016) assumed that $\tilde{g}$ is convex, which is not the case with collision or strain limits. Jiang et al. (2019); Themelis and Patrinos (2020) either assumed that A has full row rank, which implies that there is always a consensus x satisfying all the constraints as long as each constraint is satisfied in a separate manner, or they assume that $\tilde{g}$ is a real instead of an extended real function, which implies that $\tilde{g}$ cannot be used to model constraints. This is not the case with non-trivial problem instances. In the case of Figure 3, again for example, A cannot have full row rank. Further, $\tilde{g}$ is a non-convex extended real function. As summarized in Table 1, the existing ADMM is not a suitable solver for Equation 7 even if we know that the problem is strictly feasible.

Method	f	g	Additional Assumption
Deng and Yin (2016)	CVX,ER	CVX,ER	F
Jiang et al. (2019)	NC,ER	NC,R	F
Themelis and Patrinos (2020)	NC,R	NC,ER	F,S
Ours	NC,ER	BCVX,ER	F

Table 1. Assumptions on the problem data in different methods. (NC: Nonconvex, CVX: Convex, BCVX: Bi-convex, R: Real Function, ER: Extended Real Function, F: Feasible solution exists, S: A is Surjective and has full row rank) Our method allows the solution of problems where g can be non-convex, extended-real and A does not have full row rank.

Bi-convex Relaxation & BC-ADMM

To overcome the limitations of existing ADMM variants, BC-ADMM introduces a bi-convex relaxation. In the toy example shown in Figure 3, we demonstrate how the collision constraints can be reformulated using the hyperplane separation theorem, as illustrated in Figure 4.

This reformulation introduces a new set of variables representing the separating planes, which we concatenate into a vector z . While this increases the overall number of decision variables, it also induces a special structure in the problem, which forms the basis of our convergence analysis. In the case of Figure 4, we define:

$$\begin{aligned} z &= (n_{12}^T, d_{12}^T, n_{23}^T, d_{23}^T, n_{31}^T, d_{31}^T)^T \\ g(y, z) &= P_r(y_1, y_2, n_{12}, d_{12}) + \\ &\quad P_r(y_2, y_3, n_{23}, d_{23}) + \\ &\quad P_r(y_3, y_1, n_{31}, d_{31}). \end{aligned} \quad (9)$$

In summary, we consider the bi-convex variant of Equation 5 with a slack variable z as follows:

$$\operatorname{argmin}_{x, z \in \mathcal{Z}} f(x) + g(Ax, z), \quad (10)$$

where we take the following additional assumption:

Assumption 3. i) $f : \mathbb{R}^n \rightarrow \mathbb{R}^+ \cup \{\infty\}$ is continuous and differentiable in $\operatorname{dom}(f)$ and $g : \mathbb{R}^{m+l} \rightarrow \mathbb{R}^+ \cup \{\infty\}$ is continuous and twice continuously differentiable in $\operatorname{dom}(g)$. ii) $f(\bullet) + g(A\bullet, \bullet)$ is coercive. iii) A strictly feasible solution $x^0 \in \mathbb{R}^n, z^0 \in \mathcal{Z}$ is given, such that $f(x^0) + g(Ax^0, z^0) < \infty$. iv) $g(\bullet, z)$ is convex, $g(y, \bullet)$ is σ -strongly convex, and $\mathcal{Z}$ is convex and non-empty.

In other words, the non-convexity of g is encoded in the additional variable z , and g is convex otherwise. We will show that a large class of problems has such relaxations. In our BC-ADMM algorithm, we relax Equation 10 similar to the standard ADMM algorithm and introduce a slack variable y , giving the following problem:

$$\operatorname{argmin}_{x, y, z} f(x) + g(y, z) \quad \text{s.t. } Ax = y. \quad (11)$$

BC-ADMM then proceeds by introducing the following Lagrangian function:

$$\begin{aligned} \mathcal{L}(x, y, z, \lambda) \\ = f(x) + g(y, z) + \frac{\beta}{2} \|Ax - y\|^2 + \lambda^T (Ax - y). \end{aligned} \quad (12)$$

Finally, BC-ADMM updates x, y, z, λ iteratively by the following rule, with superscript denoting the iteration number:

$$\begin{cases} y^0 = Ax^0 & z^0 = z(y^0) & \lambda^0 = \nabla_y g(y^0, z^0) & (13a) \\ x^{k+1} = \operatorname{argmin}_x \mathcal{L}(x, y^k, z^k, \lambda^k) + \frac{\beta_x}{2} \|x - x^k\|^2 & (13b) \\ y^{k+1} = \operatorname{argmin}_y \mathcal{L}(x^{k+1}, y, z^k, \lambda^k) + \frac{\beta_y}{2} \|y - y^k\|^2 & (13c) \\ z^{k+1} = \operatorname{argmin}_{z \in \mathcal{Z}} \mathcal{L}(x^{k+1}, y^{k+1}, z, \lambda^k) & (13d) \\ \lambda^{k+1} = \lambda^k + \beta(Ax^{k+1} - y^{k+1}). & (13e) \end{cases}$$

Here, for Equation 13b, we assume that a locally optimal solution can be computed starting from the initial guess of x^k . For Equation 13c and Equation 13d, the locally optimal solution is also global due to convexity. Indeed, since g is σ -strongly convex in z , we know that z is the unique minimizer of the function g , which depends only on y . Due to the

uniqueness, we can write z as a function of y denoted as $z(y)$. Note that the benefit of ADMM is retained in our bi-convex relaxation. Indeed, the update of y and z can both be done by solving small problems in parallel. Taking the toy problem of Equation 3 for example, the z update corresponds to adjusting the separating plane for each collision constraint, which can be decomposed into subproblems related to (n_{12}, d_{12}) , (n_{23}, d_{23}) , and (n_{31}, d_{31}) , respectively.

Remark 4. Note that the two steps Equation 13c and Equation 13d might not be well-defined because we only assume the function g is proper as a function of (y, z) but $g(\bullet, z^k)$ or $g(y^{k+1}, \bullet)$ might not be proper, meaning that y^{k+1} and z^{k+1} might not have a finite solution. But we can easily see that BC-ADMM is well-defined by induction. As our base step, note that $g(y^0, z^0) < \infty$ by our initial condition with $z^0 \in \mathcal{Z}$. As our inductive step, suppose $g(y^k, z^k) < \infty$ with $z^k \in \mathcal{Z}$, then since Equation 13c is minimizing the objective, we have $g(y^{k+1}, z^k) < \infty$. Further, since Equation 13d is minimizing the objective, we have $g(y^{k+1}, z^{k+1}) < \infty$.

Remark 4 implies that BC-ADMM can be practically implemented without encountering infinite values or improper functions during each subproblem solve. However, we are still unclear whether solving the series of subproblems would in turn solve the problem in Equation 11, which is the main goal of our analysis. To this end, we follow the standard analysis technique and treat the subproblem solvers in Equation 13c, 13d, and 13e as oracles, while we analyze the number of BC-ADMM iterations needed until convergence. We will show that, by a proper choice of positive parameters β, β_x, β_y , each iteration generated by BC-ADMM will be strictly feasible and the algorithm converges with an oracle complexity of $O(\epsilon^{-2})$. The result is formally claimed below and proved in Section .

Theorem 5. Taking Assumption 9 and under sufficiently large β_x, β_y, β , each iteration generated by BC-ADMM satisfies $f(x^k) + g(Ax^k, z^k) < \infty$. Further, BC-ADMM converges the ϵ -stationary solution of Equation 10 with an oracle complexity of $O(\epsilon^{-2})$.

Sketch of Proof. To establish Theorem 5, we propose a novel analysis technique. Specifically, we note that, when the function g is not an extended-real function, but rather a real function with Lipschitz continuous gradient, then Equation 11 takes a similar form as the prior setting Jiang et al. (2019). Therefore, we propose to replace g with another function G , such that G is a real function with Lipschitz continuous gradient. Further, we make sure that the two functions, g and G , match up to second derivatives in a sufficiently large compact set, as illustrated in Figure 5. Such a function G is guaranteed to exist by the Whitney extension theorem Whitney (1992). With the function g replaced by G , we derive a modified BC-ADMM algorithm denoted as BC-ADMM_G, for which a slightly modified version of the analysis in Jiang et al. (2019) can be applied to establish convergence. Finally, we show that BC-ADMM_G generates a solution trajectory that lies in the compact domain where $g = G$, which establishes the convergence of BC-ADMM.

Theorem 5 extends prior ADMM algorithms to solve optimizations under a large number of constraints in the form

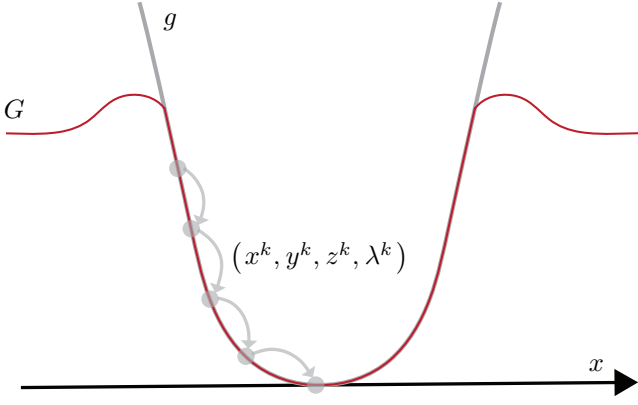


Figure 5. An illustration of our technique of analysis. We replace the extended real function g (gray) with a real function G with Lipschitz continuous gradient (red). With g replaced by G , we derive a modified BC-ADMM algorithm denoted as BC-ADMM_G . The solution sequence generated by BC-ADMM_G (gray dots and arrows), is restricted to the compact set where $g = G$, which establishes the convergence of BC-ADMM.

of soft barrier functions, while achieving the same order of convergence rate of optimizing general non-convex functions using the first-order methods Jiang et al. (2019). However, we emphasize that the parameter choices for β_x, β_y, β in our analysis are not defined in a constructive manner. In other words, we cannot design an algorithm to compute these parameters that achieves the claimed convergence speed. To tackle this difficulty, we propose a practical algorithm in the next section, which is guaranteed to converge under arbitrary choices of these parameters.

Practical Algorithm and Applications

Our analytic results require sufficiently large β_x, β_y, β , which cannot be algorithmically determined. Therefore, we need a practical algorithm that can automatically detect the parameters. In this section, we combine several approaches to detect these parameters, leading to an efficient and robust implementation of our method with an additional assumption that f is convex.

Practical Algorithm

Our practical implementation is summarized in Algorithm 1, where we adopt several approaches to ensure its robustness, anytime feasible property, and efficacy. First, to ensure that our method guarantees convergence without carefully tuning its parameters, we propose to use a lazy strategy to update z^{k+1} . Note that when z^{k+1} is fixed, i.e. Equation 13d is not executed, BC-ADMM reduces to the standard ADMM applied to convex optimization when f is convex. With z fixed, a wide range of parameters leads to convergence of x to the (globally) optimal solution of $f(\bullet) + g(A\bullet, z)$ Deng et al. (2017). Therefore, we propose to only update z^{k+1} when x^{k+1} sufficiently reduces the objective function level and the optimal solution of the convex objective is sufficiently approached. Specifically, we require the objective function value to be reduced by at least $(1 - \eta)(g(y^{k+1}, z^{k+1}) - g(y^{k+1}, z^*))$. We further require that the convergence measure Φ^{k+1} of the convex optimization has a norm less than η^K . We will show that these conditions

will be ultimately met and z^{k+1} will be updated. Further, our lazy update allows us to track a monotonically decreasing best solution denoted as $\langle x, z \rangle^*$. This strategy is outlined in Line 17. In practice, we find that BC-ADMM converges even without these conditions, so we choose to be less stringent and set η very close to 1.

Second, without sufficiently large parameters, we are not guaranteed to return a feasible solution upon early termination. To ensure the anytime feasible property, we propose to keep track of the best solution, denoted as x^*, z^* . When a solution is detected to be infeasible, we restart from x^*, z^* , while increasing β_y, β by a constant factor of κ_y and κ . Note that the factor κ we use for β is strictly larger than the factor κ_y for β_y . This is because our analysis requires β to be much larger than β_y . This strategy is outlined in Line 34. Combining the strategies above, we argue for the asymptotic convergence of Algorithm 1 in the following result, which is proved in Section .

Theorem 6. Assume that new constraints have never been detected at Line 19. Taking Assumption 9 and assuming f is convex, Algorithm 1 converges to the ϵ -stationary solution of Equation 10.

Sketch of Proof Our analysis is divided into two sections. First, we show that Line 34 would adaptively increase β_x, β_y, β . Ultimately, the assumptions of Theorem 5 will hold. In other words, these parameters will only be modified for finitely many iterations. Second, we analyze the lazy update in two cases, where z^{k+1} can be updated for finitely or infinitely many times. In both cases, we combine the uniqueness of function $z(y)$ and the sufficient decrease condition below Line 17 to establish the termination condition blow Line 12 after finitely many iterations.

Theorem 6 result immediately implies that the termination condition below Line 12 will be met within finitely many iterations. As a result, Algorithm 1 becomes a practical optimization solver that converges with a wide range of parameter choices for β_y , and β , which automatically determines β_x . Unfortunately, we are unable to prove convergence speed in this case, but instead demonstrates satisfactory performance across several applications.

Finally, we observe that in many problems, a large number of constraints co-exist. When the number of agents in Figure 3 grows, for example, we must introduce a collision constraint for each pair of agents, leading to a quadratic rate of growth in the number of terms in g with respect to the number of agents. However, many of these constraints can be ignored when the pair of agents is far apart. To effectively limit the number of constraints, we could introduce a preemptive constraint detector. Formally, we assume $g = \sum_i g_i(y, z)$ is decomposed into finitely many terms, each modeling a singleton constraint. We further introduce the function $\mathcal{V} = \text{Constraint-Detected}(x, g)$, which determines whether g_i is violated and returns the violated index set $\mathcal{V} = \{i | g_i(Ax, z) = \infty \forall z \in \mathcal{Z}, g_i \text{ not included in } g\}$. Using this function, we can start by ignoring all the constraints, and then iteratively introduce constraints preemptively detected. For example, in the case of Figure 3, the constraint detector can take the form of a collision detector Pan et al. (2012). Unfortunately, some constraint detectors can only find constraints that are already violated, but BC-ADMM

requires the problem to be strictly feasible. Therefore, we propose to revert to the previous solution when such constraints are detected, where the problem data, including f, g, A , is modified. This strategy is outlined in Line 19 and we argue for the rigorous convergence under this setting in the following result.

Corollary 7. *Taking Assumption 9 and assuming f is convex, Algorithm 1 converges to the ϵ -stationary solution of Equation 10.*

Proof. This result only extends Theorem 6 by enabling constraint detector. Note that when new constraints are detected in Line 19, we would roll back to the previous best solution $\langle x, z \rangle^*$. We claim that no constraints are violated at that configuration, because it will be rolled back to even earlier solutions otherwise. Since no constraints are violated at $\langle x, z \rangle^*$, we conclude that $g_i(Ax^*, z(Ax^*)) < \infty$ for any $i \in \mathcal{V}$. By our initialization of $z^{k+1} \leftarrow z(y^{k+1}) = z(Ax^*)$ at Line 26, we know that $g(y^{k+1}, z^{k+1}) < \infty$, so that Assumption 3 still holds and Algorithm 1 is essentially restarting with a new function g and a feasible solution. If no more constraints are detected, then the asymptotic convergence of Algorithm 1 follows from Theorem 6. Our result follows by noting that new constraints can only be detected a finite number of times.

Applications

We present several optimization problems in robotic applications that BC-ADMM can solve. To begin with, we prove the following result, which shows that if two functions g_1 and g_2 satisfy our assumption, then so does their summation $g_1 + g_2$. As a result, we can arbitrarily composite the terms discussed in this section to deal with more complex problem instances.

Corollary 8. *We suppose $g_1(y_1, z_1) \in \mathbb{R}^{m_1+l_1}$ and $\mathcal{Z}_1$ satisfy Assumption 3 iv) and vi) and $g_2(y_2, z_2) \in \mathbb{R}^{m_2+l_2}$ and $\mathcal{Z}_2$ satisfy Assumption 3 iv) and vi). Then $g = g_1 + g_2 \in \mathbb{R}^{m+l}$ with $m = m_1 + m_2$, $l = l_1 + l_2$, and $\mathcal{Z} = \mathcal{Z}_1 \times \mathcal{Z}_2$ satisfy Assumption 3 iv) and vi).*

Collision Constrained Optimization As our first application, we consider the multi-robot pose optimization problem presented in Figure 3. The key challenge in this problem lies in the handling of the collision constraint, which was originally modeled as a potential $\tilde{P}_r(y_i, y_j) = -\log_\epsilon(\|y_i - y_j\| - 2r)$. This potential is a general non-convex extended real function. However, a bi-convex potential has recently been proposed in Liang et al. (2024). As illustrated in Figure 6, we consider two convex hulls denoted as $\text{CH}(y_1, \dots, y_N)$ and $\text{CH}(y_{N+1}, \dots, y_{N+M})$, where the first hull has N vertices $y_1, \dots, y_N$ while the second hull has M vertices $y_{N+1}, \dots, y_{N+M}$. To separate the two hulls, Liang et al. (2024) introduced a separating plane denoted as $z = (n, d) \in \mathbb{R}^4$, with n being the plane normal and d being the plane offset. Put together, the potential energy preventing the two hulls

Algorithm 1 BC-ADMM

Input: x^0, A , convex f , bi-convex $g, \epsilon > 0$
Input: $\kappa > \kappa_y > 1, \gamma \in (0, 1), \eta \in (0, 1), \beta_y > 0, \beta > 0$
Output: x^k

```

1:  $\epsilon_x \leftarrow \beta_y \gamma / (\beta_y + \beta \gamma), \epsilon_y \leftarrow \beta \gamma / (\beta_y + \beta \gamma)$ 
2:  $\beta_x \leftarrow \beta(1/\epsilon_x - 1) \|A^T A\| / \gamma$ 
3:  $\langle x, z \rangle^0 \leftarrow \langle x^0, z(y^0) \rangle$ 
4:  $y^0 \leftarrow Ax^0$ 
5:  $\lambda^0 \leftarrow \nabla_y g(y^0, z^0)$ 
6:  $K \leftarrow 1$ 
7: for  $k = 0, 1, 2, \dots$  do
8:    $x^{k+1} \leftarrow \underset{x}{\operatorname{argmin}} \mathcal{L}(x, y^k, z^k, \lambda^k) + \frac{\beta_x}{2} \|x - x^k\|^2$ 
9:    $\triangleright$  The update in  $y^{k+1}$  oftentimes allow parallel solver
10:   $y^{k+1} \leftarrow \underset{y}{\operatorname{argmin}} \mathcal{L}(x^{k+1}, y, z^k, \lambda^k) + \frac{\beta_y}{2} \|y - y^k\|^2$ 
11:   $z^{k+1} \leftarrow \underset{z \in \mathcal{Z}}{\operatorname{argmin}} \mathcal{L}(x^{k+1}, y^{k+1}, z, \lambda^k)$ 
12:   $\triangleright$  Termination condition
13:  if  $\left\| \begin{pmatrix} \nabla_x f(x^{k+1}) + A^T \nabla_y g(y^{k+1}, z^{k+1}) \\ Ax^{k+1} - y^{k+1} \end{pmatrix} \right\|_\infty \leq \epsilon$ 
    then
14:    Return  $\langle x, y, z \rangle^{k+1}$ 
15:     $\Theta^{k+1} \leftarrow \left\| \begin{pmatrix} \nabla_x f(x^{k+1}) + A^T \nabla_y g(y^{k+1}, z^k) \\ Ax^{k+1} - y^{k+1} \end{pmatrix} \right\|_\infty$ 
16:     $\Lambda^{k+1} \leftarrow \frac{f(x^*) + g(Ax^*, z^*) + (1 - \eta)(g(y^{k+1}, z^{k+1}) - g(y^{k+1}, z^*))}{(1 - \eta)(g(y^{k+1}, z^{k+1}) - g(y^{k+1}, z^*))}$ 
17:     $\triangleright$  Track the best solution and lazy update  $z^{k+1}$ 
18:    if  $f(x^{k+1}) + g(Ax^{k+1}, z^{k+1}) < \Lambda^{k+1} \wedge \Theta^{k+1} \leq \eta^K$  then
19:       $\triangleright$  Handle new constraints detected
20:      if  $\emptyset \neq \mathcal{V} \leftarrow \text{Constraints-Detected}(x^{k+1}, g)$  then
21:         $\langle x, z \rangle^{k+1} \leftarrow \langle x, z \rangle^*$ 
22:         $y^{k+1} \leftarrow Ax^{k+1}$ 
23:         $\lambda^{k+1} \leftarrow \nabla_y g(y^{k+1}, z^{k+1})$ 
24:         $\triangleright$  Insert detected constraints  $\mathcal{V}$ 
25:         $g \leftarrow g + \sum_{i \in \mathcal{V}} g_i$ 
26:         $z^{k+1} \leftarrow z(y^{k+1})$ 
27:        Continue
28:      else
29:         $\langle x, z \rangle^* \leftarrow \langle x, z \rangle^{k+1}$ 
30:         $K \leftarrow K + 1$ 
31:      else
32:         $z^{k+1} \leftarrow z^k$ 
33:         $\lambda^{k+1} \leftarrow \lambda^k + \beta(Ax^{k+1} - y^{k+1})$ 
34:         $\triangleright$  Ensure feasibility by increasing  $\beta_x, \beta_y, \beta$ 
35:        if  $f(x^{k+1}) + g(Ax^{k+1}, z^{k+1}) = \infty$  then
36:           $\langle x, z \rangle^{k+1} \leftarrow \langle x, z \rangle^*$ 
37:           $y^{k+1} \leftarrow Ax^{k+1}$ 
38:           $\lambda^{k+1} \leftarrow \nabla_y g(y^{k+1}, z^{k+1})$ 
39:           $\beta_y \leftarrow \kappa_y \beta_y, \beta \leftarrow \kappa \beta$ 
40:           $\epsilon_x \leftarrow \beta_y \gamma / (\beta_y + \beta \gamma), \epsilon_y \leftarrow \beta \gamma / (\beta_y + \beta \gamma)$ 
41:           $\beta_x \leftarrow \beta(1/\epsilon_x - 1) \|A^T A\| / \gamma$ 

```

from collisions is formulated as:

$$P_r(y, z) = - \sum_{i=1}^N \log_\epsilon(y_i^T n + d - r) - \sum_{i=N+1}^{N+M} \log_\epsilon(-y_i^T n - d - r) + \frac{\sigma}{2} \|z\|^2, \quad (14)$$

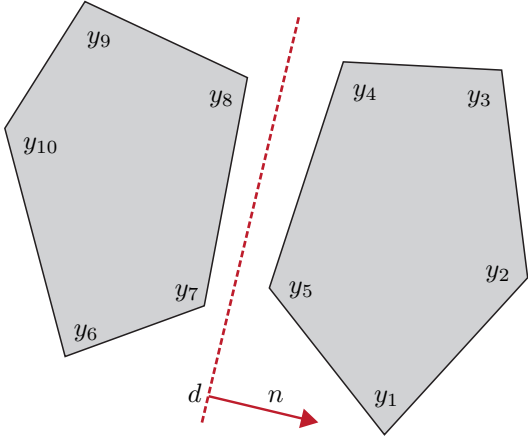


Figure 6. We illustrate the barrier potential energy to prevent collisions between a pair of convex objects, each having 5 vertices so that $N = M = 5$. The barrier energy introduces a separating plane, which is parameterized using variables $z = (n, d)$, and requires that the convex objects reside in different sides of the separating plane.

where we add a regularization $\sigma\|z\|^2/2$ to make it σ -strongly convex in z and the locally supported log-barrier function is defined as $\log_\epsilon(x) = -\max(0, (\epsilon - x)^4/x^5)$. It is easy to see that $P(y, z)$ is well-defined, bi-convex, and satisfies Assumption 3 with $\mathcal{Z} = \{n\|\|n\| \leq 1\}$. Indeed, we can show that $P_r(y, z)$ is monotonically decreasing with the length of normal vector $\|n\|$ and the additional constraint $z \in \mathcal{Z}$ enforces a unit normal $\|n\| = 1$, leading to a well-defined separating plane. Revisiting the case of Figure 3, we can replace the potential $\tilde{P}_r(y_i, y_j) = -\log_\epsilon(\|y_i - y_j\| - 2r)$ with the new potential $P_r(y, z)$ and $y = (y_i, y_j)$ to prevent collisions between the agents. However, note that the collision constraint presented here applies to more general convex objects than two points y_i, y_j , including line segments and triangles, by changing the number of vertices in each convex hull. Finally, we emphasize that if all the collision constraints are added at once, there can be a quadratic number of terms in g , significantly increasing the computational burden. To alleviate this problem, we could use a collision detector such as Pan et al. (2012) as the constraint detector in Algorithm 1.

Mass-Spring System and Strain-Limit During the simulation and control of soft robots and deformable objects, the mass-spring system is one of the most prominent and fundamental material models Huang et al. (2020); Baraff and Witkin (1998). A spring force between two particles y_i and y_j is modeled as the following potential energy:

$$\tilde{P}_{k,l}(y) = \frac{k}{2}(\|y_i - y_j\| - l)^2, \quad (15)$$

where k is the spring stiffness, l is the rest length, and $y = (y_i, y_j)$. Note that this potential energy does not satisfy our assumption because it is non-differentiable at $y_i = y_j$. However, the situation of $y_i = y_j$ is undesirable and corresponds to the configuration that the spring is infinitely compressed. Therefore, several prior works Goldenthal et al. (2007); Wang et al. (2010) propose to avoid such configurations via a hard constraint that $\|y_i - y_j\| \geq \underline{l}$ where $\underline{l}$ is the maximal allowed level of compression, which is

known as strain limiting. We can realize such constraint via the following modified spring energy with strain limiting as follows:

$$\tilde{P}_{k,l,\underline{l}}(y) = \frac{k}{2}(\|y_i - y_j\| - l)^2 - \log_\epsilon(\|y_i - y_j\| - \underline{l}). \quad (16)$$

Equipped with strain limiting, $\tilde{P}_{k,l,\underline{l}}$ becomes twice continuously differentiable in its domain. Again, $\tilde{P}_{k,l,\underline{l}}$ is a non-convex extended real function that does not satisfy our assumption. To derive a bi-convex relaxation, we follow Liu et al. (2013) to introduce a direction $z \in \mathbb{R}^3$ and define:

$$P_{k,l,\underline{l}}(y, z) = \frac{k}{2}\|y_i - y_j - zl\|^2 - \log_\epsilon(z^T(y_i - y_j) - \underline{l}), \quad (17)$$

where z is restricted to the surface unit sphere $\mathcal{Z} = \{z\|\|z\| = 1\}$. However, the set $\mathcal{Z}$ is non-convex and does not satisfy our Assumption 3. Instead, we propose a more general assumption and prove it holds for $P_{k,l,\underline{l}}(y, z)$ in Lemma 25. We further prove that the unique global minimizer of $z \in \mathcal{Z}$ is $z(y) = (y_i - y_j)/\|y_i - y_j\|$ and $P_{k,l,\underline{l}}(y, z(y)) = \tilde{P}_{k,l,\underline{l}}(y)$.

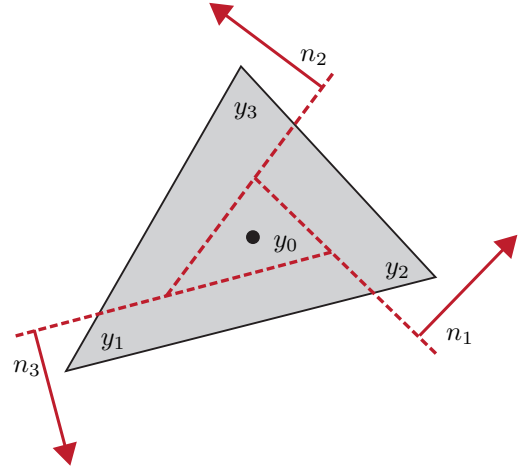


Figure 7. We illustrate the barrier potential energy to enforce strain limiting for the ARAP material model in 2D cases ($d = 2$). We first introduce an additional vertex y_0 and require that y_0 resides strictly inside the triangular element. This is achieved by introducing $d + 1 = 3$ separating planes, parameterized with normal directions $z = (n_1, n_2, n_3)$, and requiring that each edge resides in the opposite side of the corresponding separating plane to y_0 .

ARAP Deformation To simulate more complex material behaviors of soft robots and deformable objects, we need to introduce material models more general than the mass-spring system. One of the widely used material model is the ARAP elastic energy Sorkine and Alexa (2007); Kwok and Chen (2017); Fang et al. (2018). These techniques decompose a d -dimensional deformable object into a set of simplex elements. Each element is equipped with $d + 1$ vertices so that $y \in \mathbb{R}^{(d+1)d}$ and a deformable gradient $F(y) \in \mathbb{R}^{d \times d}$. The ARAP energy is then formulated as:

$$\tilde{P}_F(y) = \|F(y) - R(F(y))\|_F^2, \quad (18)$$

where $R(F(y))$ is the rotational part of the polar decomposition of $F(y)$. Intuitively, the ARAP energy encourages the soft robot to undergo a purely rigid transformation dictated by $R(F)$, and induces elastic energy otherwise. Similar to the case with mass-spring models, we can also introduce the idea of strain limiting. Let us denote $\sigma_k(F)$ as the k th singular value of F . If any $\sigma_k(F)$ is close to zero, then the corresponding element is significantly compressed along a right singular vector, which is unlikely to happen in real-world scenarios. To prevent these situations from happening, a well-known energy is the NeoHookean elastic potential Kim et al. (2012), which is formulated as:

$$\tilde{P}_F(y) = \|F(y) - R(F(y))\|_F^2 - \log \det(F(y)) + \sum_{i=1}^d \iota_{\sigma_k(F) > 0}(F), \quad (19)$$

where the additional $\iota_{\sigma_k(F) > 0}$ functions ensure that F has all positive singular values and the $-\log \det(F(y))$ term ensures that the potential function is continuous. Unfortunately, the NeoHookean does not have a bi-convex relaxation. Instead, we propose a novel biconvex relaxation to prevent $\sigma_k(F)$ from getting negative as illustrated in Figure 7, following a similar idea as Liang et al. (2024). We introduce an additional point $y_0 \in \mathbb{R}^d$ as a decision variable. We further introduce $d + 1$ separating plane normals n_i to separate y_0 and each of the $d + 1$ surfaces of the element. Mathematically, we define:

$$P_{F,\epsilon}(y, z) = \|F - R\|_F^2 + \sum_{i=1}^d \iota_{\sigma_k(F) > 0}(F) + \iota_{\text{CH}(y_1, \dots, y_{d+1})}(y_0) + \sum_{i=1}^{d+1} \frac{\sigma}{2} \|n_i\|^2 - \sum_{i=1}^{d+1} \sum_{0 < j \neq i} \log_{\epsilon}(n_i^T(y_j - y_0)), \quad (20)$$

where the additional $\iota_{\text{CH}(y_1, \dots, y_{d+1})}(y_0)$ term ensures that y_0 belongs to the simplicial complex, the regularization $\sigma \|n_i\|^2 / 2$ ensures $P_{F,\epsilon}$ is σ -strongly convex in n_i . Finally, we re-define $y = (y_0, \dots, y_{d+1}) \in \mathbb{R}^{(d+2)d}$ and $z = (n_1, \dots, n_{d+1}, R)$ and $\mathcal{Z} = \prod_{i=1}^{d+1} \{n_i \|n_i\| \leq 1\} \times \text{SO}(3)$. It is easy to see that $P_{F,\epsilon}(y, z)$ is a biconvex energy. Again, since $\mathcal{Z}$ is not a convex set, it does not satisfy our Assumption 3. We use the more general assumption and prove that our $P_{F,\epsilon}(y, z)$ satisfies it in Lemma 27. Note that the subproblems in Equation 13c and Equation 13d of BC-ADMM become easy to solve, where Equation 13c amounts to a small convex optimization, while Equation 13d amounts to small convex optimizations and a polar decomposition.

Numerical Stability

Compared with general-purpose infeasible point constrained optimizers Gill et al. (2005); Wachter (2002), the higher numerical stability and robustness are the main benefits of ADMM-type algorithms. Indeed, when the constraint qualifications (nearly) fail, i.e., when the Jacobian of the constraint functions is nearly singular, the QP subproblem can be ill-conditioned. Instead, stable and robust algorithms exist for each and every subproblem of BC-ADMM.

Specifically, Equation 13b corresponds to an unconstrained optimization of a sufficiently smooth function, for which either gradient descend or Newton's method can be used with proper globalization techniques such as line-search methods. Equation 13c corresponds to a strongly convex optimization, for which robust algorithms have been well discussed in Boyd and Vandenberghe (2004). Equation 13d amounts to compute the function $z(y^{k+1})$. If Assumption 3 holds, then Equation 13d again amounts to a strongly convex optimization. If our more general Assumption 9 holds, then the stability of solving Equation 13d needs to be analyzed in a case-by-case manner. Fortunately, our function $z(y^{k+1})$ admits closed-form solutions in all the practical problems. In the case of the modified spring energy $P_{k,l,l}(y, z)$, for example, the closed-form global minimizer is $z(y) = (y_i - y_j) / \|y_i - y_j\|$. In the case of the modified ARAP energy with strain limiting $P_{F,\epsilon}(y, z)$, the rotation matrix R can be computed via the polar decomposition.

Evaluation

We use C++ to implement Algorithm 1 and OpenMP to solve yz -subproblems in parallel. In all our examples, f is always a quadratic function, so that the x -subproblem amounts to a linear system solve for which we use the sparse direct solver in the Eigen library. For all our examples, we choose $\kappa_y = 2$, $\kappa = 2.1$, $\gamma = 0.95$, $\eta = 1 - 10^{-5}$, and run experiments on a single desktop machine with the AMD Ryzen Threadripper 3970X CPU. The remaining parameters β_y and β need to be tuned for each problem, which is a standard treatment in practical implementations Overby et al. (2017); Ni et al. (2022). We evaluate our method on a row of four benchmarks listed below, where we compare our method with two standard baselines: Gradient Descent (GD) and Newton's Method. As a major difference from BC-ADMM, these algorithms solve Equation 10 without decoupling f and g . Therefore, we only need to update two variables, x and z , in an alternative manner. In GD, the simple first-order update rule for x is used:

$$x \leftarrow x - \alpha_{\text{GD}} [\nabla_x f(x) + A^T \nabla_y g(Ax, z)], \quad (21)$$

with α_{GD} being the sufficiently small step size found by the line search procedure to ensure a strict function value decrease. The Newton's method is defined similarly, but uses the following second-order update rule:

$$x \leftarrow x - \left[\nabla_x^2 f(x) + A^T \nabla_y^2 g(Ax, z) A + \alpha_{\text{Newton}} I \right]^{-1} [\nabla_x f(x) + A^T \nabla_y g(Ax, z)]. \quad (22)$$

Here α_{Newton} is the sufficiently large regularization found by the line search procedure to ensure a strict function value decrease. After updating x , both GD and Newton's method update z by setting $z \leftarrow z(Ax)$. Typically, Newton's method achieves faster convergence, but each iteration is slower by requiring solving a large-scale sparse linear system. In our implementation, we use the state-of-the-art sparse linear solver Chen et al. (2008) to compute the second-order search direction. Finally, both methods suffer from the drawback of having to search for a global step size α_{GD} and α_{Newton} that can slow down the progress of the overall optimization.

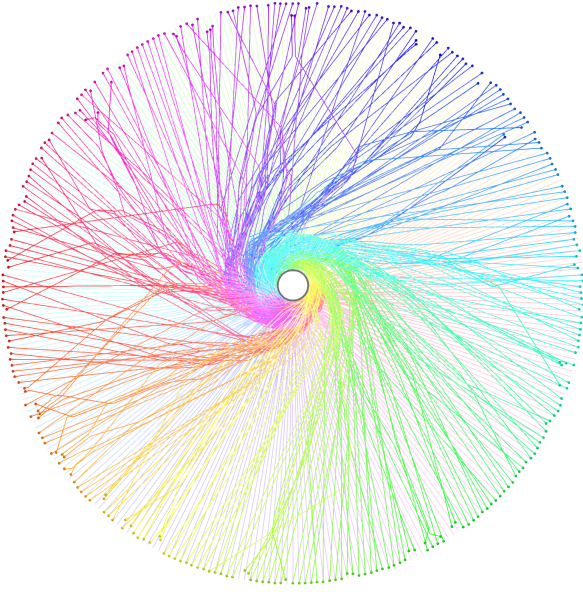


Figure 8. We plan motions for $N = 300$ robots with an obstacle in the middle and plot the trajectories computed by BC-ADMM.

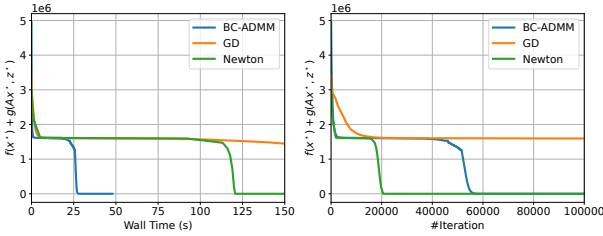


Figure 9. Objective function level for multi-agent navigation, plotted against wall time (left) and iteration number (right). All three solves would have the robot congested in the middle of the circle, forming a congested configuration. This corresponds to the plateau in the middle of convergence curve. After a while, BC-ADMM and Newton's method find way for all robots to reach the other end.

Multi-Agent Navigation We solve the large-scale multi-agent navigation problem Karamouzas et al. (2017); Weiss et al. (2017). We notice that we can compute collision-free navigation trajectories by solving a robot pose optimization problem as illustrated in Figure 3. Specifically, for a set of N robots $x = (x_1, x_2, \dots, x_N)$, we adopt the following setup:

$$\begin{aligned}
 f(x) &= \sum_{i=1}^N \underbrace{\frac{1}{2} \|x_i - x_i^*\|^2}_{\text{goal-reaching}} \\
 g(Ax, z) &= \sum_{i=1}^N \sum_{j=i+1}^N \underbrace{P_r((x_i, x_j), z_{ij})}_{\text{self-collision}} + \sum_{o_l=1}^N \sum_{i=1}^N \underbrace{P_r((x_i, o_l), z_{il})}_{\text{obstacle-collision}}, \quad (23)
 \end{aligned}$$

where x_i^* is the goal of the i th robot, r is the robot radius, and P_r is the collision potential between a pair of geometric entities. Besides robot-robot collisions, there are static obstacles in the environment (the l th obstacle denoted as o_l), and we use two sets of collision potential terms

to prevent their interpenetrations. The potential number of collision terms is quadratic, which leads to a large number of yz -subproblems to solve. To accelerate computation, we use a 2D collision checker as our constraint detector, where we only include P_r terms in our optimization when the distance between two agents is less than $(2 + \epsilon_{\text{margin}})r$ with ϵ_{margin} set to a margin distance. We emphasize that, although the above formulation only solves for a single pose for each robot, the optimizer would generate a series of intermediary poses during each iteration. These intermediary poses can be connected into a robot trajectory. Specifically, let us denote $k(i)$ as the index of the subset of iterations where the two conditions below Line 17 of Algorithm 1 are satisfied, i.e. $z^{k(i)+1}$ is updated to $z(y^{k(i)+1})$ at the end of the $k(i)$ th iteration. After the optimization, we reconstruct the i th robot trajectory as $(x_i^0, x_i^{k(1)+1}, x_i^{k(2)+1}, \dots)$. We claim that these trajectories are valid, collision-free trajectories navigating robots towards their goal x_i^* . To see this is the case, note that z^{k+1} is not updated between iterations $k(i-1)$ and $k(i)$. Since the second condition below Line 17 is satisfied at $k(i)$ th iteration, we have $\Theta^{k(i)+1} < \infty$ and thus:

$$f(x^{k(i)+1}) + g(Ax^{k(i)+1}, z^{k(i-1)+1}) < \infty. \quad (24)$$

Since the first condition below Line 17 is satisfied at $k(i-1)$ th iteration, we have:

$$f(x^{k(i-1)+1}) + g(Ax^{k(i-1)+1}, z^{k(i-1)+1}) < \infty. \quad (25)$$

By the definition of P_r , the above two conditions imply that robots at configuration $x^{k(i-1)+1}$ and $x^{k(i)+1}$ are both collision-free. Further, in both of these configurations, the pair of robots x_i and x_j is separated by the same plane $z_{ij}^{k(i-1)+1}$. Now, since the plane-separated regions are convex regions and linear interpolated points of a convex region stay inside this region, we conclude that any linear interpolated robot configurations between $x^{k(i-1)+1}$ and $x^{k(i)+1}$ are also collision-free. This proves that the piecewise linear robot trajectories $(x_i^0, x_i^{k(1)+1}, x_i^{k(2)+1}, \dots)$ are valid, collision-free paths. In conclusion, we have shown that BC-ADMM is a suitable solver for multi-agent navigation problems. As illustrated in Figure 8 we conduct experiments on a standard benchmark with $N = 300$ robots arranged in a circle with an obstacle in the middle, where each robot has their goal on the opposite side of the circle. All three solvers would generate similar robot trajectories, where robots are first congested in the middle and then push each other to find ways to reach the goals. The convergence history of all three algorithms is plotted in Figure 9. Our method converges after only 30 seconds of computation. In comparison, GD cannot make sufficient progress even after 150 seconds of computation. Instead, Newton's method takes a much smaller number of iterations to converge. But since each iteration is much more costly, their overall computational cost is higher. Finally, we compare the performance of a variant of Algorithm 1 with the check for the two conditions below Line 17 turned off. In other words, we always update $z^{k+1} \leftarrow z(y^{k+1})$. In this case, there is no guarantee that the algorithm converges, unless we use the parameters β_x, β_y, β satisfying Theorem 5 that are difficult to estimate. Using the same set of practical

parameters, the performance of these two variants of BC-ADMM is profiled in Figure 10. The result shows that this variant of BC-ADMM can frequently violate the constraints, leading to the energy value being ∞ . Therefore, we cannot recover collision-free trajectories for the robots, rendering this variant of BC-ADMM an inappropriate solver for multi-agent navigation problems.

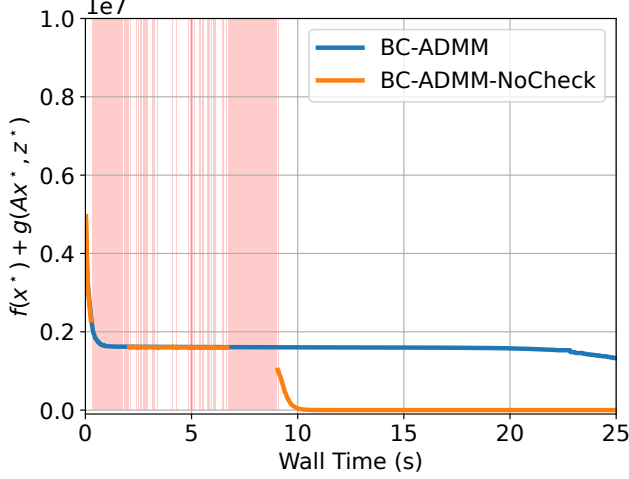


Figure 10. We run multi-agent navigation using standard Algorithm 1 (BC-ADMM) and a variant of Algorithm 1 with the check for the two conditions below Line 17 turned off (BC-ADMM-NoCheck). From the convergence history, we can see that, in many intermediary iterations, BC-ADMM-NoCheck violates the constraints, leading to the energy value being ∞ , as marked by the vertical red lines. As a result, we cannot reconstruct collision-free trajectories for the robots, rendering BC-ADMM-NoCheck an inappropriate solver for multi-agent navigation problems.

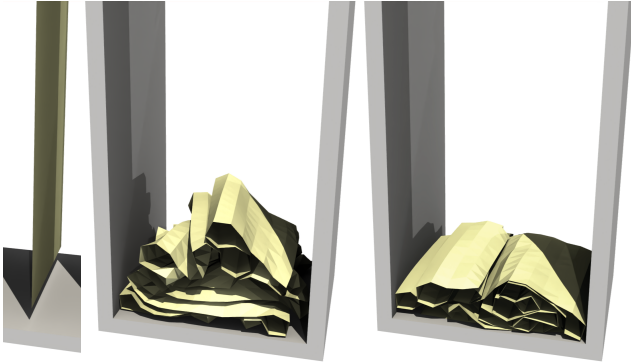


Figure 11. We drop a flat piece of cloth (left) on the ground and use BC-ADMM to compute its final configuration under a high (middle) and low (right) cloth thickness.

Quasistatic Deformable Object Simulation Deformable object simulation is a foundational module in robotic manipulation tasks Lin et al. (2022). Thanks to our reformulated mass-spring material model $P_{k,l,l}$, our BC-ADMM can be used as a deformable object simulator. Although ADMM algorithms have been used as physical simulators in prior works Overby et al. (2017); Ouyang et al. (2020). However, Overby et al. (2017) used a soft penalty function to handle constraints, which is not guaranteed to resolve all interpenetrations. This is a well-known challenge

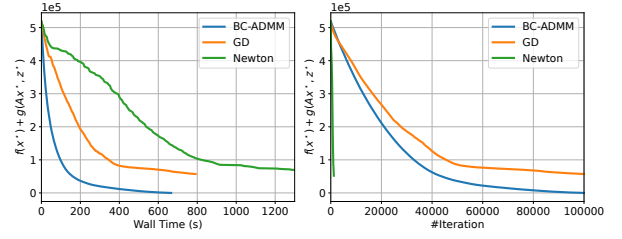


Figure 12. Objective function level for deformable object simulation, plotted against wall time (left) and iteration number (right). When BC-ADMM computes the ultimate pose for the piece of cloth, the two other solvers are still far from convergence.

in cloth simulation Harmon et al. (2009); Li et al. (2020) where inter-penetrations are notoriously hard to resolve. Similarly, Ouyang et al. (2020) did not consider any collision constraints or strain limits. Instead, our novel analysis and algorithm design allow all the constraints to be incorporated, making BC-ADMM a full-fledged and robust physical simulator.

In our second benchmark, we consider a quasistatic simulation problem where a long flat piece of cloth is dropped into a box and our goal is to compute the ultimate configuration of the cloth. This problem is formulated as a potential energy minimization problem. We represent the cloth as a triangle mesh with x being the concatenated vertex positions, and we denote each triangle of the mesh as consisting of three vertices, denoted x_i, x_j, x_k . We minimize the combined gravitational energy $f(x) = -x^T M g$ with M being the mass matrix and g being the gravitational acceleration. Further, the term g is a summation over mass-spring energy terms $P_{k,l,l}$ and collision potential terms P_r . Specifically, we model the box as a set of convex box-shaped obstacles. Between a pair of obstacle (the l th obstacle denoted as o_l) and the triangle x_i, x_j, x_k , we introduce a collision potential P_r to prevent collision between the two geometric primitives. Another set of collision potential P_r is introduced to prevent self-collisions. To model the material of cloth, we introduce two types of springs as done in Liu et al. (2013), the 1-ring and 2-ring spring models, which model the stretch and bending resistance of the cloth, respectively. Note that in this benchmark, we do not have $f \neq 0$, which violates our assumptions. But since the piece of cloth is confined to a box, we know that f cannot go to $-\infty$, i.e. it is lower-bounded. Therefore, we can simply add a constant to f to make it positive. Put together, our objective function is formulated as:

$$\begin{aligned}
 f(x) &= \underbrace{-x^T M g}_{\text{gravitational-force}} + \text{const.} \\
 g(Ax, z) &= \sum_{\langle x_{ijk}, o_l \rangle} \underbrace{P_r((x_i, x_j, x_k, o_l), z_{ijkl})}_{\text{obstacle-collision}} + \\
 &\quad \sum_{\langle x_{ijk}, x_{lmn} \rangle} \underbrace{P_r((x_i, x_j, x_k, x_l, x_m, x_n), z_{ijklmn})}_{\text{self-collision}} + \\
 &\quad \sum_{x_{ij}} \underbrace{P_{k,l,l}((x_i, x_j), z_{ij})}_{\text{mass-spring}}.
 \end{aligned} \tag{26}$$

Our benchmark involves a mesh with 1111 vertices, i.e. $x \in \mathbb{R}^{3333}$. Our formulation allows us to simulate a piece of cloth with different thickness, since the parameter r in $P_r(y, z)$ can be modeled as the thickness parameter and the

solutions under a high- and a low-thickness are illustrated in Figure 11. The convergence history of all three algorithms are summarized in Figure 12. In this setting, the iterative cost of Newton’s method is much higher than the two other solvers, so that the first method outperforms. Again, BC-ADMM is the fastest to converge.



Figure 13. We drape three pieces of cloth with different extensible limit. From left to right, we use $(\bar{l} - l)/l = 0.1, 0.2, 0.4$.

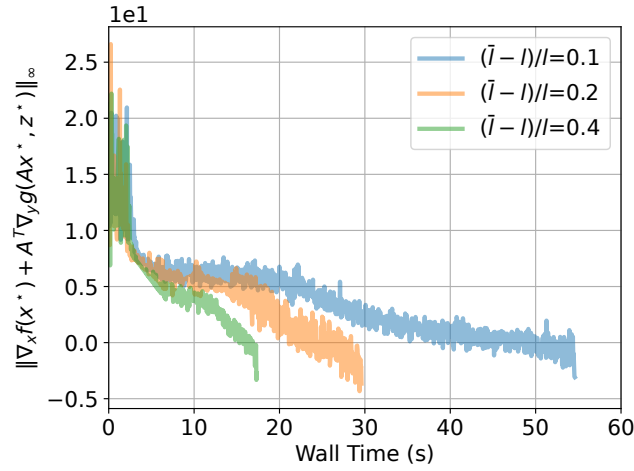


Figure 14. With a more stringent extensible limit, BC-ADMM experiences slower convergence.

We further show that we can slightly extend our formulation to simulate nearly inextensible cloth Goldenthal et al. (2007), thus allowing our method to incorporate a wider range of material models. An inextensible cloth has a stretch limit of $\bar{l} > l$ such that a spring between y_i and y_j has the maximal length of $\bar{l}$. This behavior can be implemented by an additional energy term denoted as $P_{\bar{l}}(y) = -\log_{\epsilon}(\bar{l} - \|y_i - y_j\|)$. This energy term is convex in y and thus trivially bi-convex, which is compatible with our assumption. Further, we note that $P_{\bar{l}}(y)$ is not differentiable and has a non-smooth singularity when $y_i = y_j$. But due to the strain limit in the mass-spring energy $P_{k,l,\bar{l}}(y, z)$, we know that $\|y_i - y_j\| \geq \bar{l}$, so that this singularity is removable. Combined, the inextensible mass-spring potential is denoted as $P_{k,l,\bar{l},\bar{l}}(y, z) = P_{k,l,\bar{l}}(y, z) + P_{\bar{l}}(y)$. In Figure 13, we drape three pieces of cloth with different extensible limits. As shown in Figure 14, a more stringent extensible limit leads to slower convergence, which is consistent with the observation in Goldenthal et al. (2007).

UAV Trajectory Optimization In this benchmark, we solve the classical problem of optimizing a smooth and collision-free UAV trajectory Gao et al. (2018); Ni et al. (2021)

by minimizing the trajectory-integrated jerk under various constraints. Specifically, a UAV trajectory is represented by a piecewise Bèzier curve with each curve represented as a 5th-order polynomial. Consecutive Bèzier curves are C^2 -smoothly connected with matching derivatives up to second order. We represent all the Bèzier curves using their control points concatenated into a vector x , where we use A_{ij} to denote the ij th block of matrix A that extracts the j th control points of the i th Bèzier curve. Similarly, A_i is the i th block of matrix A that extracts all the 6 control points of the curve. There are obstacles in the environment, each represented using a convex object. To prevent the Bèzier curve from intersecting the obstacle, we introduce a collision potential P_r for each pair of Bèzier curve ($A_i x$) and convex obstacle o_l . Our collision potential follows a similar idea as Ni et al. (2021) that ensures all the control points and the obstacle lay on two different sides of the separating plane. Now since the actual curve is bounded by the convex hull of the control points, our collision potential ensures the entire curve is collision-free. Again, we use a collision detector to progressively introduce collision terms into the optimizer. We further introduce velocity and acceleration limits to ensure that the trajectory is executable. Specifically, to introduce a velocity upper bound of $\bar{v}$, we compute the derivative of each curve and extract its control points, with the j th control point of the i th derivative curve’s denoted as $A'_{ij} x$ and we bound all the control points to be within the range $\bar{v}$ by introducing the log-barrier function $-\log_{\epsilon}(\bar{v} - \sqrt{\|A'_{ij} x\|^2 + \epsilon})$, where we introduce a small regularization ϵ to ensure the function is smooth. A similar technique can be used to adopt the acceleration bound. Put together, we use the following problem definition:

$$f(x) = \underbrace{\frac{1}{2} x^T J x}_{\text{trajectory-integrated jerk}}$$

$$g(Ax, z) = \sum_{\langle A_i, o_l \rangle} \underbrace{P_r((A_i x, o_l), z_{il})}_{\text{obstacle-collision}} + \sum_{A'_{ij}} \underbrace{-\log_{\epsilon}(\bar{v} - \sqrt{\|A'_{ij} x\|^2 + \epsilon})}_{\text{velocity-bound}}, \quad (27)$$

where J is the Hessian matrix of the trajectory-integrated jerk. Our formulation is the same as Ni et al. (2021), which is more robust than prior work Gao et al. (2018) by ensuring collision-free properties using log-barrier functions P_r . However, it is known that the log-barrier functions lead to slow convergence. Therefore, Ni et al. (2022) proposed to accelerate the solver by using ADMM. Unfortunately, their formulation does not allow splitting the collision potential terms P_r into sub-problems, i.e., all the P_r terms are treated as part of f instead of g , which requires their solver to handle all the collision potentials together and use a line-search to upper bound the overall step size. Instead, our formulation allows a fully decoupled treatment where P_r terms are handled separately using a y -subproblem. As illustrated in Figure 15, we optimize a very complex trajectory that has the UAV flying through a series of circle-shaped obstacles of a varying radius, where the UAV trajectory is represented using 393 Bèzier curves. The convergence history of all three algorithms is plotted in Figure 16 in terms of energy

level and gradient norm. In this example, BC-ADMM has almost the same performance as Newton's method in terms of wall time, which is much faster than GD. However, since each iteration of BC-ADMM is much faster than Newton's method, our algorithm is more amenable to early stop and returns a solution with moderately high quality, which is a common practice in the community of UAV trajectory optimization. From the plot of the gradient norm, we can see that BC-ADMM is the best performer. Although both BC-ADMM and GD are first-order algorithms, GD exhibits very noisy gradients, while BC-ADMM consistently reduces the gradient norm.

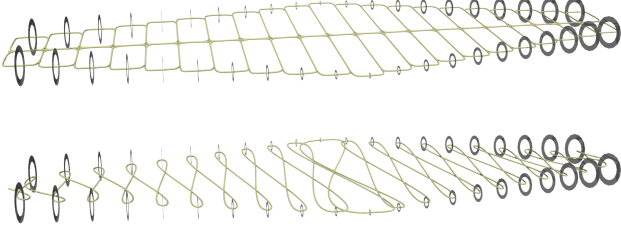


Figure 15. We optimize a complex UAV trajectory where the UAV flies through a series of circle-shaped obstacles of varying radius. The top figure shows our initial trajectory, and the bottom figure shows the optimized configuration.

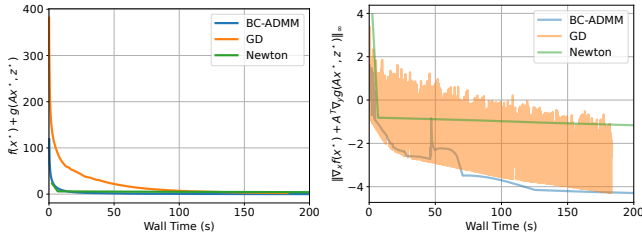


Figure 16. We plot the convergence history of the three algorithms in optimizing the UAV trajectory, where the left figure shows the energy value and the right figure shows the gradient norm. BC-ADMM has similar performance as Newton's method, significantly outperforms GD.

Soft Robot Gripper Simulation We solve the problem proposed in Fang et al. (2018), where a soft robot is actuated by cables and serves as fingers of a gripper, treating ADMM as a quasistatic soft robot simulator. In our problem setting, the robot is discretized using a finite element mesh, and the internal elastic energy is discretized using our modified, bi-convex ARAP energy $P_{F,\epsilon}$. To model the cable-driven actuator, we notice that a cable segment connecting two points x_i and x_j can be exactly modeled as a convex potential energy:

$$P_{\text{cable},a}(x_i, x_j) = a\sqrt{\|x_i - x_j\|^2 + \epsilon}, \quad (28)$$

where ϵ is a small regularization to ensure smoothness and a is the positive constant representing the internal tension force magnitude in the cable. It can be verified that the negative gradient of $P_{\text{cable},a}$ is the force vector with magnitude a and pointing along the line segment between x_i and x_j . As usual, all the collisions are handled using our collision potential term P_r , and we use collision detectors to progressively

introduce these potential terms. Put together, our problem setup is summarized below:

$$\begin{aligned} f(x) &= \underbrace{-x^T M g}_{\text{gravitational-force}} + \text{const.} \\ g(Ax, z) &= \sum_{\langle x_{ijk}, o_l \rangle} \underbrace{P_r((x_i, x_j, x_k, o_l), z_{ijkl})}_{\text{obstacle-collision}} + \\ &\quad \sum_{\langle x_{ijk}, x_{lmn} \rangle} \underbrace{P_r((x_i, x_j, x_k, x_l, x_m, x_n), z_{ijklmn})}_{\text{self-collision}} + \\ &\quad \sum_{x_{ijkl}} \underbrace{P_{F,\epsilon}((x_i, x_j, x_k, x_l), z_{ijkl})}_{\text{ARAP-material-model}} + \\ &\quad \sum_{x_{ij}} \underbrace{P_{\text{cable},a}(x_i, x_j)}_{\text{cable-driven actuator}}. \end{aligned} \quad (29)$$

In our benchmark, we model the soft robot gripper to grasp another soft ball, where the ball is modeled using ARAP material as well. We setup the ball to have a much smaller stiffness than the finger (we use 100 : 1 stiffness ratio), so that the fingers can impose large grasping forces on the soft ball, forming a firm grip and causing extreme deformations in the ball. Starting from a trivial initialization, our goal is to compute the ultimate pose of the fingers and the soft ball. This is a challenging benchmark where large forces need to be transferred from the gripper to the ball, through collision potential terms P_r . Further, the extreme deformations can cause complex collisions between objects, and any violations of collision constraints can lead to infinite energy values and failure of optimization. Again, we show the initial and optimized configurations of this benchmark in Figure 18 and we plot the convergence history in Figure 19. Our BC-ADMM significantly outperforms the two other algorithms in terms of energy value and gradient norm. Thanks to our modified ARAP energy with strain limiting, BC-ADMM can simulate extremely large deformations without inverted elements as compared with the standard ARAP formulation, as illustrated in Figure 17.

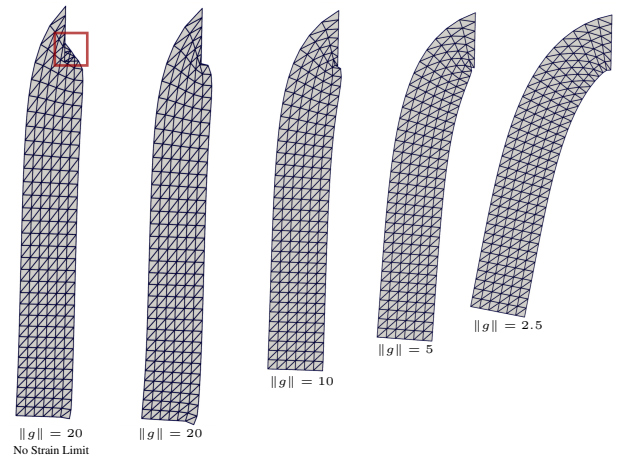


Figure 17. We simulate a soft beam under different gravitational coefficients (marked below) and compare our modified ARAP energy $P_{F,\epsilon}$ with the standard ARAP energy without strain limiting by setting $P_F = \|F - R\|_F^2$. Under a large gravitational force $\|g\| = 20$, the result without strain limiting exhibits an inverted element (red box), while our ARAP energy with strain limiting always predicts faithful deformation results.

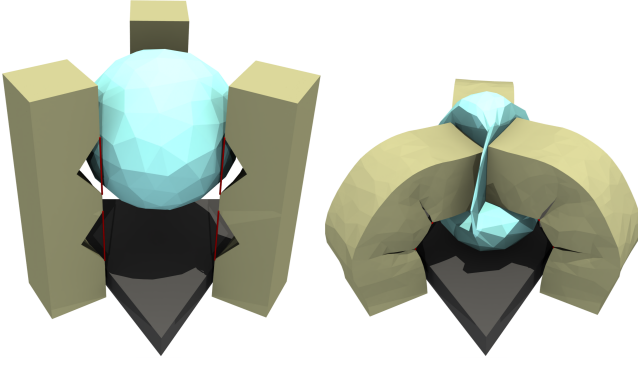


Figure 18. We simulate a 3-fingered, cable-actuated (red line segment) soft robot gripper (brown) to grasp a soft-ball (blue), the initial and optimized configuration are shown on the left and right, respectively. The ball is much softer than the gripper, so that the gripper can impose extreme forces and cause significant deformations.

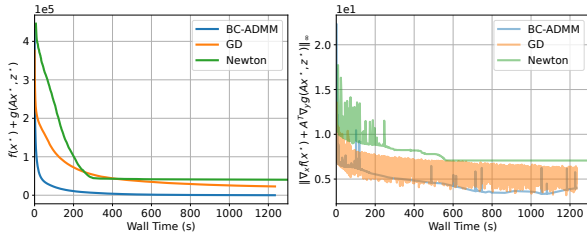


Figure 19. We plot the convergence history of the three algorithms in the soft robot gripper simulation benchmark, where the left figure shows the energy value and the right figure shows the gradient norm. BC-ADMM outperforms the two other algorithms in both criteria.

Scalability

Empirically, larger problem instances lead to an accumulation of log-barrier terms, which increases the numerical stiffness of the optimization problem and results in a greater number of iterations to reach convergence. We examine how computational cost scales with problem size across four benchmark tasks. While it is common practice in robotics applications to terminate optimization after a fixed number of iterations, such comparisons only capture per-iteration cost and fail to reflect true convergence behavior. Instead, we evaluate the total computational cost required for convergence. To do so, we first run the optimizer for a large number of iterations to ensure full convergence and obtain a reference solution, denoted by (x^{**}, z^{**}) . We then define convergence as the point at which the relative objective value is within 1% of this reference:

$$\frac{|[f(x^*) + g(Ax^*, z^*)] - [f(x^{**}) + g(Ax^{**}, z^{**})]|}{|f(x^{**}) + g(Ax^{**}, z^{**})|} < 1\%.$$

This criterion is sufficient for nearly all practical applications, as we observe no measurable differences in robot behavior or object deformation once the threshold is met. Our findings are summarized in Figure 20. In some tasks, such as multi-agent navigation and UAV trajectory optimization, we even observe a decrease in computational cost with increasing problem size. Nevertheless, the overall scaling

trend across all tasks remains approximately linear with respect to problem size.

We further analyzed the scalability of all three methods using the example shown in Figure 21. In this benchmark, a piece of cloth is dropped onto a grid of needles—a standard scenario for evaluating the robustness and scalability of cloth simulation algorithms. By varying the resolution of the cloth mesh, we profile the convergence behavior of all three methods, as illustrated in Figure 22. Our results show that the performance advantage of our method becomes more pronounced at higher resolutions. This is primarily due to the fact that the per-iteration complexity of BC-ADMM scales approximately linearly with problem size, while the direct matrix factorization required by Newton’s method scales superlinearly. Additionally, the global line-search strategies employed by both Newton’s method and gradient descent tend to become more restrictive as the problem size increases. Finally, we note the inherent challenge in ensuring fair comparisons across varying problem sizes, as different solvers may converge to different local solutions. Furthermore, a rigorous theoretical characterization of the scalability of ADMM-based optimization methods remains an open problem in the field.

Parameter Sensitivity

Our main result in Assumption 3 indicates that the optimizer can be sensitive to all three parameters: β_x , β_y , and β . In practice, however, we find that Algorithm 1 provides a robust mechanism for automatically tuning β_x and β_y to ensure convergence. In all experiments, we initialize $\beta_y = 1$ and allow the algorithm to adaptively adjust both β_x and β_y during optimization. In contrast, the algorithm remains sensitive to the global penalty parameter β , which requires manual tuning across different benchmarks. This sensitivity is consistent with observations in prior work, such as Overby et al. (2017). In Figure 23, we compare convergence behavior for two examples under varying values of β . The results reveal the existence of a sweet spot—where the optimizer converges most efficiently—and demonstrate degraded convergence when β is set either too high or too low. For deformable object simulation, we recommend following the empirical guideline proposed in Overby et al. (2017), setting β to the square root of the average material stiffness, i.e., $\beta = \sqrt{k}$ in $P_{k,l,l}$. However, we note that the stiffness of the log-barrier function can vary dynamically depending on the proximity between objects and obstacles. Therefore, for other applications such as multi-agent navigation and UAV trajectory optimization, we simply use a fixed value of $\beta = 100$.

Conclusion

In this paper, we introduce BC-ADMM, a new optimizer that can solve non-convex constrained optimization problems. Our theoretical analysis shows that BC-ADMM has convergence speed guarantees and practical convergence guarantees in an asymptotic sense. We further show that BC-ADMM can be used in a wide range of robotic applications, including multi-agent navigation, UAV trajectory optimization, deformable object simulation, and soft robot simulation. Our numerical experiments show that,

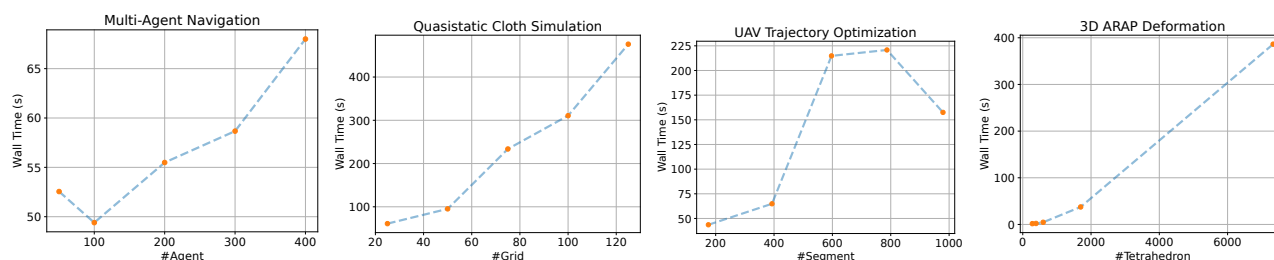


Figure 20. We plot the computational cost as a function of problem size across four tasks. For the multi-agent navigation problem, we use the agent setup shown in Figure 8, where a varying number of robots are initialized in a circular configuration. In the quasistatic cloth simulation task, following the setup in Figure 11, we vary the cloth size by increasing the number of rectangular grid cells along the vertical dimension. For the UAV trajectory optimization task, based on the configuration in Figure 15, we vary the number of circular obstacles. As the number of obstacles increases, the UAV trajectory is subdivided into a corresponding number of Bèzier curve segments. Lastly, for ARAP deformation, we adopt the 3D version of the setup in Figure 17, where we vary problem size by discretizing the 3D beam using different numbers of tetrahedral elements.

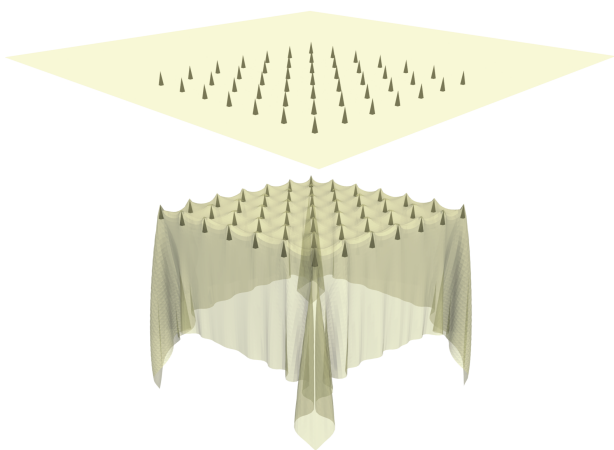


Figure 21. The initial (top) and final (bottom) iteration during the quasistatic optimization of a piece of cloth dropped onto a grid of needles.

when solving large-scale problem instances, BC-ADMM outperforms both gradient descent and Newton’s method in terms of wall clock time. However, our formulation still suffers from several limitations. First, our theoretical analysis does not guarantee a convergence speed of the practical Algorithm 1. Instead, our convergence speed guarantee requires sufficiently large parameters that are hard to estimate. Second, our method heavily relies on a bi-convex relaxation of non-convex constraints, which potentially limits its range of applications in other robotic problems. An important class of robotic applications is the optimization under dynamic constraints, which has been investigated in other structured optimizers Stella et al. (2017); Tang and Daoutidis (2019). Incorporating dynamic constraints in BC-ADMM poses a promising direction of future research. Finally, our practical Algorithm 1 requires a centralized check to determine whether variables can be updated, which prevents BC-ADMM from solving very large-scale problems in a decentralized manner. We hope these limitations inspire future research to improve our techniques.

References

Araki H and Yamagami S (1981) An inequality for hilbert-schmidt norm. *Communications in Mathematical Physics* 81(1): 89–96.

- Baraff D and Witkin A (1998) Large steps in cloth simulation. In: *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH ’98*. New York, NY, USA: Association for Computing Machinery. ISBN 0897919998, p. 43–54. DOI:10.1145/280814.280821. URL <https://doi.org/10.1145/280814.280821>.
- Birgin EG and Martinez JM (2008) Improving ultimate convergence of an augmented lagrangian method. *Optimization Methods and Software* 23(2): 177–195.
- Boggs PT and Tolle JW (1995) Sequential quadratic programming. *Acta numerica* 4: 1–51.
- Boyd S, Parikh N, Chu E, Peleato B and Eckstein J (2011) Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning* 3(1): 1–122.
- Boyd SP and Vandenberghe L (2004) *Convex optimization*. Cambridge university press.
- Byravan A, Boots B, Srinivasa SS and Fox D (2014) Space-time functional gradient optimization for motion planning. In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 6499–6506.
- Chang TH, Hong M and Wang X (2014) Multi-agent distributed optimization via inexact consensus admm. *IEEE Transactions on Signal Processing* 63(2): 482–497.
- Chen Y, Davis TA, Hager WW and Rajamanickam S (2008) Algorithm 887: Cholmod, supernodal sparse cholesky factorization and update/downdate. *ACM Transactions on Mathematical Software (TOMS)* 35(3): 1–14.
- Conn AR, Gould G and Toint PL (2013) *LANCELOT: a Fortran package for large-scale nonlinear optimization (Release A)*, volume 17. Springer Science & Business Media.
- Deits R and Tedrake R (2015) Efficient mixed-integer planning for uavs in cluttered environments. In: *2015 IEEE international conference on robotics and automation (ICRA)*. IEEE, pp. 42–49.
- Deng W, Lai MJ, Peng Z and Yin W (2017) Parallel multi-block admm with $\mathcal{O}(1/k)$ convergence. *Journal of Scientific Computing* 71: 712–736.
- Deng W and Yin W (2016) On the global and linear convergence of the generalized alternating direction method of multipliers. *Journal of Scientific Computing* 66: 889–916.
- Dontchev AL and Rockafellar RT (2009) *Implicit functions and solution mappings*, volume 543. Springer.

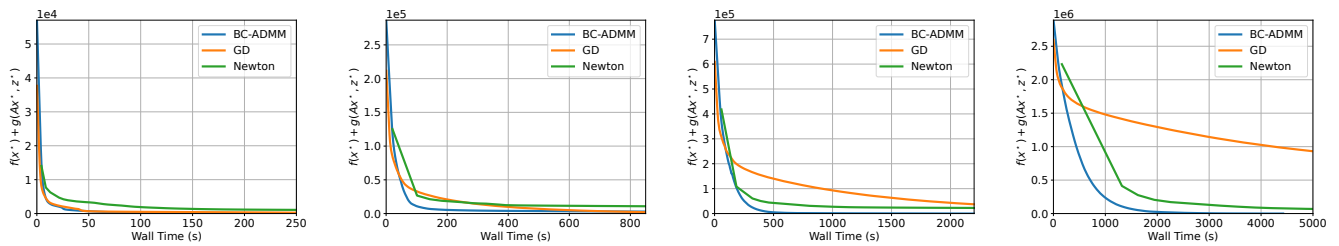


Figure 22. The convergence history of the three algorithms in the example of Figure 21. From left to right, we use a cloth resolution of 30^2 , 60^2 , 90^2 , and 120^2 vertices. Our results show that the performance advantage of our method becomes more pronounced at higher resolutions.

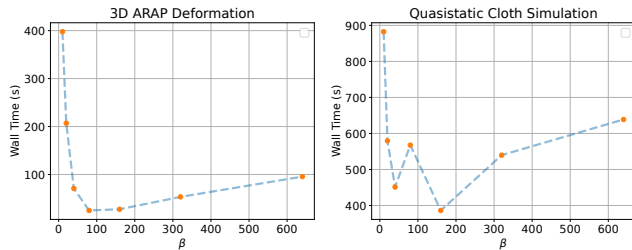


Figure 23. We profile the cost to converge under a wide range of parameter choices for two of our benchmarks.

Du W, Yao S, Wang X, Xu Y, Xu W and Lu C (2024) Intersection-free robot manipulation with soft-rigid coupled incremental potential contact. *IEEE Robotics and Automation Letters* 9(5): 4487–4494. DOI:10.1109/LRA.2024.3381012.

Duriez C (2013) Control of elastic soft robots based on real-time finite element method. In: *2013 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 3982–3987.

Fang G, Matte CD, Kwok TH and Wang CC (2018) Geometry-based direct simulation for multi-material soft robots. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 4194–4199.

Fu XM and Liu Y (2016) Computing inversion-free mappings by simplex assembly. *ACM Transactions on Graphics (TOG)* 35(6): 1–12.

Gao F, Wu W, Lin Y and Shen S (2018) Online safe trajectory generation for quadrotors using fast marching method and bernstein basis polynomial. In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 344–351. DOI:10.1109/ICRA.2018.8462878.

Gill PE, Murray W and Saunders MA (2005) Snopt: An sqp algorithm for large-scale constrained optimization. *SIAM review* 47(1): 99–131.

Goldenthal R, Harmon D, Fattal R, Bercovier M and Grinspun E (2007) Efficient simulation of inextensible cloth. *ACM Trans. Graph.* 26(3): 49–es. DOI:10.1145/1276377.1276438. URL <https://doi.org/10.1145/1276377.1276438>.

Guy SJ, Chhugani J, Kim C, Satish N, Lin M, Manocha D and Dubey P (2009) Clearpath: highly parallel collision avoidance for multi-agent simulation. In: *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. pp. 177–187.

Harmon D, Vouga E, Smith B, Tamstorf R and Grinspun E (2009) Asynchronous contact mechanics. *ACM Trans. Graph.* 28(3). DOI:10.1145/1531326.1531393. URL <https://doi.org/10.1145/1531326.1531393>.

Huang W, Huang X, Majidi C and Jawed MK (2020) Dynamic simulation of articulated soft robots. *Nature communications* 11(1): 2233.

Hughes RL (2002) A continuum theory for the flow of pedestrians. *Transportation Research Part B: Methodological* 36(6): 507–535.

Jiang B, Lin T, Ma S and Zhang S (2019) Structured nonconvex and nonsmooth optimization: algorithms and iteration complexity analysis. *Computational Optimization and Applications* 72(1): 115–157.

Karaman S, Walter MR, Perez A, Frazzoli E and Teller S (2011) Anytime motion planning using the rrt*. In: *2011 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 1478–1483. DOI:10.1109/ICRA.2011.5980479.

Karamouzias I, Sohre N, Narain R and Guy SJ (2017) Implicit crowds: Optimization integrator for robust crowd simulation. *ACM Transactions on Graphics (TOG)* 36(4). DOI:10.1145/3072959.3073705.

Khatana V and Salapaka MV (2022) Dc-distadmm: Admm algorithm for constrained optimization over directed graphs. *IEEE Transactions on Automatic Control* 68(9): 5365–5380.

Kim B, Lee SB, Lee J, Cho S, Park H, Yeom S and Park SH (2012) A comparison among neo-hookean model, mooney-rivlin model, and ogden model for chloroprene rubber. *International Journal of Precision Engineering and Manufacturing* 13: 759–764.

Kovalsky SZ, Aigerman N, Basri R and Lipman Y (2015) Large-scale bounded distortion mappings. *ACM Transactions on Graphics (TOG)* 34(6): 191–1.

Kwok TH and Chen Y (2017) Gdfe: Geometry-driven finite element for four-dimensional printing. *Journal of Manufacturing Science and Engineering* 139(11): 111006.

Li M, Ferguson Z, Schneider T, Langlois TR, Zorin D, Panozzo D, Jiang C and Kaufman DM (2020) Incremental potential contact: intersection-and inversion-free, large-deformation dynamics. *ACM Transactions on Graphics (TOG)* 39(4): 49.

Liang C, Gao X, Wu K and Pan Z (2024) Second-order convergent collision-constrained optimization-based planner. *IEEE Robotics and Automation Letters*.

Lin X, Huang Z, Li Y, Tenenbaum JB, Held D and Gan C (2022) Diffskill: Skill abstraction from differentiable physics for deformable object manipulations with tools. In: *International Conference on Learning Representations (ICLR)*. p. 0.

Lipson H (2014) Challenges and opportunities for design, simulation, and fabrication of soft robots. *Soft Robotics* 1(1): 21–27.

- Liu T, Bargteil AW, O'Brien JF and Kavan L (2013) Fast simulation of mass-spring systems. *ACM Transactions on Graphics (TOG)* 32(6): 1–7.
- Ni R, Pan Z and Gao X (2022) Robust multi-robot trajectory optimization using alternating direction method of multiplier. *IEEE Robotics and Automation Letters* 7(3): 5950–5957.
- Ni R, Schneider T, Panozzo D, Pan Z and Gao X (2021) Robust & asymptotically locally optimal uav-trajectory generation based on spline subdivision. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 7715–7721.
- Nocedal J and Wright SJ (1999) *Numerical optimization*. Springer.
- Ouyang W, Peng Y, Yao Y, Zhang J and Deng B (2020) Anderson acceleration for nonconvex admm based on douglas-rachford splitting. In: *Computer Graphics Forum*, volume 39. Wiley Online Library, pp. 221–239.
- Overby M, Brown GE, Li J and Narain R (2017) Admm $\supseteq$ projective dynamics: Fast simulation of hyperelastic models with dynamic constraints. *IEEE Transactions on Visualization and Computer Graphics* 23(10): 2222–2234. DOI:10.1109/TVCG.2017.2730875.
- Pan J, Chitta S and Manocha D (2012) Fcl: A general purpose library for collision and proximity queries. In: *2012 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 3859–3866.
- Pan Z, Chen T, Fang X, Bretl T, Gao X and Wu K (2022) Environment warped gait trajectory optimization for complex terrains. *IEEE Robotics and Automation Letters* 7(4): 12403–12410.
- Qin L, Peng H, Huang X, Liu M and Huang W (2024) Modeling and simulation of dynamics in soft robotics: A review of numerical approaches. *Current Robotics Reports* 5(1): 1–13.
- Reynolds CW (1987) Flocks, herds and schools: A distributed behavioral model. In: *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*. pp. 25–34.
- Rockafellar RT and Wets RJB (2009) *Variational analysis*, volume 317. Springer Science & Business Media.
- Rus D and Tolley MT (2015) Design, fabrication and control of soft robots. *Nature* 521(7553): 467–475.
- Saravanos AD, Tsolovikos A, Bakolas E and Theodorou E (2021) Distributed Covariance Steering with Consensus ADMM for Stochastic Multi-Agent Systems. In: *Proceedings of Robotics: Science and Systems*. Virtual, p. 0. DOI:10.15607/RSS.2021.XVII.075.
- Schulman J, Duan Y, Ho J, Lee A, Awwal I, Bradlow H, Pan J, Patil S, Goldberg K and Abbeel P (2014) Motion planning with sequential convex optimization and convex collision checking. *The International Journal of Robotics Research* 33(9): 1251–1270.
- Sorkine O and Alexa M (2007) As-rigid-as-possible surface modeling. In: *Symposium on Geometry processing*, volume 4. Citeseer, pp. 109–116.
- Stella L, Themelis A, Sopasakis P and Patrinos P (2017) A simple and efficient algorithm for nonlinear model predictive control. In: *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE, pp. 1939–1944.
- Tang W and Daoutidis P (2019) Distributed nonlinear model predictive control through accelerated parallel admm. In: *2019 American Control Conference (ACC)*. pp. 1406–1411. DOI: 10.23919/ACC.2019.8814732.
- Themelis A and Patrinos P (2020) Douglas–rachford splitting and admm for nonconvex optimization: Tight convergence results. *SIAM Journal on Optimization* 30(1): 149–181.
- Van den Berg J, Lin M and Manocha D (2008) Reciprocal velocity obstacles for real-time multi-agent navigation. In: *2008 IEEE international conference on robotics and automation (ICRA)*. IEEE, pp. 1928–1935.
- Wachter A (2002) *An interior point algorithm for large-scale nonlinear optimization with applications in process engineering*. Carnegie Mellon University.
- Wang H, O'Brien J and Ramamoorthi R (2010) Multi-resolution isotropic strain limiting. *ACM Transactions on Graphics (TOG)* 29(6): 1–10.
- Wang Y, Yin W and Zeng J (2019) Global convergence of admm in nonconvex nonsmooth optimization. *Journal of Scientific Computing* 78(1): 29–63.
- Wang Z, Zhou X, Xu C, Chu J and Gao F (2020) Alternating minimization based trajectory generation for quadrotor aggressive flight. *IEEE Robotics and Automation Letters* 5(3): 4836–4843.
- Weiss T, Litteneker A, Jiang C and Terzopoulos D (2017) Position-based multi-agent dynamics for real-time crowd simulation. In: *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. pp. 1–2.
- Whitney H (1992) Analytic extensions of differentiable functions defined in closed sets. *Hassler Whitney Collected Papers* : 228–254.
- Xu Y and Yin W (2013) A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion. *SIAM Journal on imaging sciences* 6(3): 1758–1789.
- Zhang D, Liang C, Gao X, Wu K and Pan Z (2024) Provably feasible semi-infinite program under collision constraints via subdivision. *IEEE Transactions on Robotics* 40: 2602–2619. DOI:10.1109/TRO.2024.3391649.
- Zhang J, Peng Y, Ouyang W and Deng B (2019) Accelerating admm for efficient simulation and optimization. *ACM Transactions on Graphics (TOG)* 38(6): 1–21.
- Zhang X, Liniger A and Borrelli F (2020) Optimization-based collision avoidance. *IEEE Transactions on Control Systems Technology* 29(3): 972–983.
- Zhou B, Gao F, Wang L, Liu C and Shen S (2019) Robust and efficient quadrotor trajectory generation for fast autonomous flight. *IEEE Robotics and Automation Letters* 4(4): 3529–3536.

Convergence Analysis

We need a novel technique to analyze the convergence of BC-ADMM. Our main idea is to construct an Lyapunov candidate and establish its descendant property. However, such analysis requires the function g to have Lipschitz continuous gradient. To work around this obstacle, we propose to use Whitney extension theorem Whitney (1992) and construct a function G with Lipschitz-continuous gradient. We then consider BC-ADMM with g replaced by G and denote the algorithm as BC-ADMM_G (Section) and establish its convergence by Lyapunov analysis (Section). Finally, we show that the two algorithms, BC-ADMM and BC-ADMM_G, generate the same sequence (Section).

BC-ADMM_G

To construct the modified algorithm BC-ADMM_G, we introduce the following result to prove the existence of G . Specifically, for a function twice differentiable on a compact set of $\mathbb{R}^{m+l}$, we can extend it to a function that is twice differentiable in the entire $\mathbb{R}^{m+l}$. Further, we can slightly modify this function to equip it with Lipschitz continuous gradient. Let us define the set:

$$\mathcal{S}(r, M) = \{(y, z) \mid g(y, z) \leq M \wedge \|(y, z)\| \leq r\}$$

$$\mathcal{S}_{\mathcal{Z}}(r, M) = \{(y, z) \mid g(y, z) \leq M \wedge \|(y, z)\| \leq r \wedge z \in \mathcal{Z}\}.$$

We aim to prove a theorem that is slightly more general theorem based on a weaker assumption on the function g as follows, which allows us to solve more problem instances:

Assumption 9. i) $f: \mathbb{R}^n \rightarrow \mathbb{R}^+ \cup \{\infty\}$ is continuous and differentiable in $\text{dom}(f)$ and $g: \mathbb{R}^{m+l} \rightarrow \mathbb{R}^+ \cup \{\infty\}$ is continuous and twice continuously differentiable in $\text{dom}(g)$. ii) $f(\bullet) + g(A\bullet, \bullet)$ is coercive. iii) A strictly feasible solution $x^0 \in \mathbb{R}^n, z^0 \in \mathcal{Z}$ is given, such that $f(x^0) + g(Ax^0, z^0) < \infty$. iv) $g(\bullet, z)$ is convex. vi) For $(y, z) \in \mathcal{S}_{\mathcal{Z}}(r, M)$, $g(y, \bullet) + \iota_{\mathcal{Z}}(\bullet)$ has a unique global minimizer on the closed set $\mathcal{Z}$ denoted as a function $z(y)$, such that $(y, z(y)) \in \mathcal{S}_{\mathcal{Z}}(\bar{r}_{zy}^{r, M}, M)$. Further for $(y, z(y)) \in \mathcal{S}_{\mathcal{Z}}(r, M)$, $z(y)$ is an $L_{zy}^{r, M}$ -Lipschitz continuous function.

We first present a set of preparatory results:

Lemma 10. We take Assumption 9. Then for any $r > 0$ and $M > 0$, g has the Lipschitz continuous gradient on $\mathcal{S}(r, M)$ with a modulus $L_g^{r, M} < \infty$.

Proof. For each $(y, z) \in \mathcal{S}(r, M)$, we can find an open neighborhood $\mathcal{G}_{y,z}(\rho(y, z))$ with radius $\rho(y, z)$ such that $\|\nabla^2 g(y', z')\| \leq L(y, z) < \infty$ for $(y', z') \in \mathcal{G}_{y,z}(\rho(y, z))$. Now since $\mathcal{S}(r, M)$ is compact, we can find a finite sub-cover and we choose $L_g^{r, M}$ to be the maximum over these finite modulus $L(y, z)$.

Lemma 11. We take Assumption 9. Then for any $r > 0$ and $N > M > 0$, there exists some $\delta^{r, M, N} > 0$, such that for any $(y, z) \in \mathcal{S}(r, M)$, $\mathcal{G}_{y,z}(\delta^{r, M, N}) \subseteq \mathcal{S}(r + \delta^{r, M, N}, N)$.

Proof. Consider another set $\mathcal{S}'(r + 1, N) = \{g(y, z) \leq N \wedge \|(y, z)\| \leq r + 1\}$. Clearly, $\mathcal{S}(r, M)$ and $\mathcal{S}'(r + 1, N)$ are both compact and disjoint because g is continuous, so the distance between them is a positive value denoted as $\delta^{r, M, N, \star} > 1$. We can show that $\delta^{r, M, N} = \min(\delta^{r, M, N, \star}, 1)$ satisfies our needs.

Lemma 12. We take Assumption 9. Suppose $(y, z) \in \mathcal{S}_{\mathcal{Z}}(r, M)$ and $\|y' - y\| \leq \delta^{r, M, N}$, then $\|z(y) - z(y')\| \leq L_{zy}^{r, M, N} \|y' - y\|$ for some $L_{zy}^{r, M, N} < \infty$.

Proof. For any point y'' on the line segment between y and y' , we have $(y'', z) \in \mathcal{S}_{\mathcal{Z}}(r + \delta^{r, M, N}, N)$ by Lemma 11, so by our Assumption 9 vi), $(y'', z(y'')) \in \mathcal{S}_{\mathcal{Z}}(\bar{r}_{zy}^{r, M, N}, M)$ where we define $\bar{r}_{zy}^{r, M, N} = r + \delta^{r, M, N}$. Further by our Assumption 9 vi), $z(y'')$ restricted to this set is Lipschitz continuous with a modulus of $L_{zy}^{\bar{r}_{zy}^{r, M, N}, M}$ re-defined as $L_{zy}^{r, M, N}$, which is our desired result.

Lemma 13. We take Assumption 9. Then for any $r, M > 0$ such that $\mathcal{S}(r, M)$ is not empty, we can define a twice-differentiable real function $G_{r, M}: \mathbb{R}^{m+l} \rightarrow \mathbb{R}$ such that: i) $G_{r, M}$ and g match up to second derivatives on $\mathcal{S}(r, M)$; ii) $G_{r, M} = 0$ when $\|(y, z)\| \geq 2r$; iii) $G_{r, M}$ has $L_G^{r, M}$ -Lipschitz continuous gradient with $L_g^{r, M} \leq L_G^{r, M} < \infty$; iv) $G_{r, M} \geq \underline{G}_r > -\infty$.

Proof. Clearly $\mathcal{S}(r, M)$ is compact, so the Whitney extension theorem results in a twice-differentiable function $G_{r, M}^*$ such that $G_{r, M}^*$ and g match up to second derivatives on $\mathcal{S}(r, M)$. We then define a smooth decay function:

$$\eta_r(d) = \begin{cases} 1 & d \leq r \\ (2r - x)^3(6x^2 - 9rx + 4r^2)/r^5 & d \in (r, 2r) \\ 0 & d \geq 2r \end{cases}.$$

Finally, we can satisfy all the conditions by defining: $G_{r, M}(y, z) = \eta_r(\|(y, z)\|)G_{r, M}^*(y, z)$. Indeed, i) and ii) are trivial. To establish iii), note that $G_{r, M}$ has $L_g^{r, M}$ -Lipschitz continuous gradient on $\mathcal{S}(r, M)$ with $L_g^{r, M} < \infty$ by Lemma 10, $G_{r, M}$ has 0-Lipschitz continuous gradient when $\|(y, z)\| \geq 2r$. The remaining area is bounded. Similarly, to establish iv), note that $G_{r, M} = g \geq 0$ on $\mathcal{S}(r, M)$, $G_{r, M} = 0$ when $\|(y, z)\| \geq 2r$, and the remaining area is bounded.

The main benefit of analyzing the extended function $G_{r, M}$ instead of g lies in the fact that $G_{r, M}$ has $L_G^{r, M}$ -Lipschitz continuous gradient, so that we have the following convenient result in the entire Euclidean space:

$$\|\nabla G_{r, M}(y, z) - \nabla G_{r, M}(y', z')\| \leq L_G^{r, M} \|(y, z) - (y', z')\|, \quad (30)$$

and:

$$\begin{aligned} & G_{r, M}(y', z') \\ & \leq G_{r, M}(y, z) + \nabla G_{r, M}(y, z)^T((y', z') - (y, z)) + \frac{L_G^{r, M}}{2} \|(y, z) - (y', z')\|^2. \end{aligned} \quad (31)$$

We are now ready to define our modified BC-ADMM_G algorithm that aims to minimize the objective with g replaced by $G_{r, M}$. First, since $f(\bullet) + g(A\bullet, \bullet)$ is coercive by Assumption 9, we must have that all the (x, z) satisfying $f(x) + g(Ax, z) \leq M$ is bounded in the hyperball with radius R^M , i.e., $\|(x, z)\| \leq R^M$. With this notation, we assume the following parameter choice:

Assumption 14. The following parameters are used for BC-ADMM_G:

$$\begin{cases} M = 5(f(x^0) + g(Ax^0)) \\ r^1 = R^{2M/5}, r^2 = r^1 + \delta^{r^1, 2M/5, 3M/5} \end{cases} \quad (32a)$$

$$\begin{cases} r^3 = r^2 + \delta^{r^2, 3M/5, 4M/5}, r^4 = r^3 + L_{zy}^{r^2, 3M/5, 4M/5} \delta \\ \delta = \min(\delta^{r^1, 2M/5, 3M/5}, \delta^{r^2, 3M/5, 4M/5}, \delta^{r^4, 4M/5, M}) \end{cases} \quad (32b)$$

$$\begin{cases} r = r^4 + \delta^{r^4, 4M/5, M} \\ \beta_x \geq \frac{2\|A\|\|\nabla f(x^0) + A^T \nabla_y g(Ax^0, z^0)\|}{\delta} + L_f \end{cases} \quad (32c)$$

$$\begin{cases} \beta_x \geq \frac{4(2M/5 - \underline{G}_{r,M})\|A\|^2}{\delta^2} \\ \beta_y \geq \frac{4(2M/5 - \underline{G}_{r,M})}{\delta^2} \end{cases} \quad (32d)$$

$$\begin{cases} \beta \geq 2L_G^{r,M} \\ \beta \geq \frac{32\beta_y^2 + 16\beta_y^2(L_G^{r,M} L_{zy}^{r^2, 3M/5, 4M/5})^2 + 16(L_G^{r,M})^2}{\beta_y} \end{cases} \quad (32e)$$

$$\begin{cases} \beta \geq \frac{2(\beta_y - L_G^{r,M})}{\beta_y + L_G^{r,M} - 2} \\ \beta \geq \frac{\delta^2}{f(x^0) + g(Ax^0, z^0)} \end{cases} \quad (32f)$$

$$\begin{cases} \beta \geq \frac{2(\beta_y - L_G^{r,M})}{\beta_y + L_G^{r,M} - 2} \\ \beta \geq \frac{\delta^2}{f(x^0) + g(Ax^0, z^0)} \end{cases} \quad (32g)$$

$$\begin{cases} \beta \geq \frac{2(\beta_y - L_G^{r,M})}{\beta_y + L_G^{r,M} - 2} \\ \beta \geq \frac{\delta^2}{f(x^0) + g(Ax^0, z^0)} \end{cases} \quad (32h)$$

$$\begin{cases} \beta \geq \frac{2(\beta_y - L_G^{r,M})}{\beta_y + L_G^{r,M} - 2} \\ \beta \geq \frac{\delta^2}{f(x^0) + g(Ax^0, z^0)} \end{cases} \quad (32i)$$

$$\begin{cases} \beta \geq \frac{2(\beta_y - L_G^{r,M})}{\beta_y + L_G^{r,M} - 2} \\ \beta \geq \frac{\delta^2}{f(x^0) + g(Ax^0, z^0)} \end{cases} \quad (32j)$$

$$\begin{cases} \beta \geq \frac{2(\beta_y - L_G^{r,M})}{\beta_y + L_G^{r,M} - 2} \\ \beta \geq \frac{\delta^2}{f(x^0) + g(Ax^0, z^0)} \end{cases} \quad (32k)$$

It is easy to see that the parameters in Assumption 14 can be satisfied by choosing $M, \delta, r, \beta_x, \beta_y, \beta$ in order to satisfy all the conditions. Next, we introduce the following Lagrangian function:

$$\begin{aligned} \mathcal{L}_{r,M}(x, y, z, \lambda) \\ = f(x) + G_{r,M}(y, z) + \frac{\beta}{2}\|Ax - y\|^2 + \lambda^T(Ax - y). \end{aligned} \quad (33)$$

Our BC-ADMM_G proceeds to update x, y, z, λ iteratively by the following rule, with superscript denoting the iteration number:

$$\begin{cases} y^0 = Ax^0 \quad z^0 = z(y^0) \quad \lambda^0 = \nabla_y g(y^0, z^0) \end{cases} \quad (34a)$$

$$\begin{cases} x^{k+1} = \underset{x}{\operatorname{argmin}} \mathcal{L}_{r,M}(x, y^k, z^k, \lambda^k) + \frac{\beta_x}{2}\|x - x^k\|^2 \end{cases} \quad (34b)$$

$$\begin{cases} y^{k+1} = \underset{y}{\operatorname{argmin}} \mathcal{L}_{r,M}(x^{k+1}, y, z^k, \lambda^k) + \frac{\beta_y}{2}\|y - y^k\|^2 \end{cases} \quad (34c)$$

$$\begin{cases} z^{k+1} = \operatorname{CLIP}(z(y^{k+1}), z^k, L_{zy}^{r^2, 3M/5, 4M/5}\|y^{k+1} - y^k\|) \end{cases} \quad (34d)$$

$$\begin{cases} \lambda^{k+1} = \lambda^k + \beta(Ax^{k+1} - y^{k+1}). \end{cases} \quad (34e)$$

We make two changes in BC-ADMM_G. First, we cannot compute the globally optimal solution in Equation 34c because $G_{r,M}(\bullet, z)$ is not convex in general, so we only assume that a locally optimal solution is computed starting from the initial guess of y^k . Second, note that we cannot reuse equation Equation 13d with g replaced by $G_{r,M}$ because $G_{r,M}(y, \bullet)$ is no longer convex in general. Therefore, we introduce a clip function defined as:

$$\operatorname{CLIP}(z(y^{k+1}), z^k, l) =$$

$$\begin{cases} z^k & \text{otherwise} \\ z(y^{k+1}) & \text{if } \begin{cases} \|z(y^{k+1}) - z^k\| \leq l \text{ and} \\ G_{r,M}(y^{k+1}, z(y^{k+1})) \leq G_{r,M}(y^{k+1}, z^k) \end{cases} \end{cases}.$$

Note that, unlike BC-ADMM, we cannot use mathematical induction to show that $g(y^{k+1}, \bullet)$ is proper in BC-ADMM_G, so we have to consider the case where $z(y^{k+1})$ does not exist and our CLIP function simply set $z^{k+1} = z^k$ in this case. Finally, similar to BC-ADMM, we assume that Equation 34b and Equation 34c is solved by local gradient descend starting from the initial guess of x^k . By the first-order optimality condition and the definition of CLIP, we have at every iteration:

$$0 = \beta_x(x^{k+1} - x^k) + \nabla f(x^{k+1}) + \beta A^T(Ax^{k+1} - y^k) + A^T \lambda^k \quad (35)$$

$$= \beta_x(x^{k+1} - x^k) + \nabla f(x^{k+1}) + A^T \lambda^{k+1} + \beta A^T(y^{k+1} - y^k) \quad (36)$$

$$0 = \beta_y(y^{k+1} - y^k) + \nabla_y G_{r,M}(y^{k+1}, z^k) - \lambda^{k+1} \quad (37)$$

$$l \geq \|\operatorname{CLIP}(a, b, l) - b\|. \quad (38)$$

In the next section, we would construct Lyapunov candidate for BC-ADMM_G and establish its descent property.

Lyapunov Analysis of FT-ADMM_G

In this section, we analyze the convergence by constructing a Lyapunov candidate in a similar fashion as Jiang et al. (2019). We start from a convenient lemma:

Lemma 15. Taking Assumption 9, the update of λ^k in FT-ADMM_G satisfies the following inequality:

$$\begin{aligned} \|\lambda^{k+1} - \lambda^k\|^2 &\leq 4(\beta_y^2 + (L_G^{r,M})^2)\|y^{k+1} - y^k\|^2 + \\ &4\beta_y^2(1 + (L_G^{r,M} L_{zy}^{r^2, 3M/5, 4M/5})^2)\|y^k - y^{k-1}\|^2. \end{aligned} \quad (39)$$

Proof. This result is derived by the following inequality:

$$\begin{aligned} &\|\lambda^{k+1} - \lambda^k\|^2 \\ &\stackrel{(1)}{\leq} 4\|\beta_y(y^{k+1} - y^k)\|^2 + 4\|\beta_y(y^k - y^{k-1})\|^2 + \\ &4\|\nabla_y G_{r,M}(y^{k+1}, z^k) - \nabla_y G_{r,M}(y^k, z^k)\|^2 + \\ &4\|\nabla_y G_{r,M}(y^k, z^k) - \nabla_y G_{r,M}(y^k, z^{k-1})\|^2 \\ &\stackrel{(2)}{\leq} 4(\beta_y^2 + (L_G^{r,M})^2)\|y^{k+1} - y^k\|^2 + 4\beta_y^2\|y^k - y^{k-1}\|^2 + \\ &4\beta_y^2(L_G^{r,M})^2\|z^k - z^{k-1}\|^2 \\ &\stackrel{(3)}{\leq} 4(\beta_y^2 + (L_G^{r,M})^2)\|y^{k+1} - y^k\|^2 + 4\beta_y^2\|y^k - y^{k-1}\|^2 + \\ &4\beta_y^2(L_G^{r,M} L_{zy}^{r^2, 3M/5, 4M/5})^2\|y^k - y^{k-1}\|^2. \end{aligned}$$

Here (1) is derived using Equation 37, (2) is due to Lemma 13, and (3) is due to Lemma 12 and Equation 38.

Next, we propose the following Lyapunov function:

$$\begin{aligned} \Psi_{r,M}(x^k, y^k, z^k, \lambda^k, y^{k-1}) &= \mathcal{L}_{r,M}(x^k, y^k, z^k, \lambda^k) + \\ &\frac{4\beta_y^2(1 + (L_G^{r,M} L_{zy}^{r^2, 3M/5, 4M/5})^2)}{\beta}\|y^k - y^{k-1}\|^2, \end{aligned}$$

and establish its monotonically decreasing property:

Lemma 16. Taking Assumption 9 and Equation 32i, then we have for FT-ADMM_G:

$$\begin{aligned} &\frac{\beta_x}{4}\|x^{k+1} - x^k\|^2 + \frac{\beta_y}{4}\|y^{k+1} - y^k\|^2 \\ &\leq \Psi_{r,M}(x^k, y^k, z^k, \lambda^k, y^{k-1}) - \\ &\Psi_{r,M}(x^{k+1}, y^{k+1}, z^{k+1}, \lambda^{k+1}, y^k). \end{aligned} \quad (40)$$

Proof. From Equation 34b and Equation 34c, we have:

$$\begin{aligned} & \mathcal{L}_{r,M}(x^{k+1}, y^{k+1}, z^k, \lambda^k) + \\ & \frac{\beta_y}{2} \|y^{k+1} - y^k\|^2 + \frac{\beta_x}{2} \|x^{k+1} - x^k\|^2 \\ & \leq \mathcal{L}_{r,M}(x^{k+1}, y^k, z^k, \lambda^k) + \frac{\beta_x}{2} \|x^{k+1} - x^k\|^2 \\ & \leq \mathcal{L}_{r,M}(x^k, y^k, z^k, \lambda^k). \end{aligned} \quad (41)$$

Now from Equation 41, Equation 34e, and Lemma 15, we have:

$$\begin{aligned} & \mathcal{L}_{r,M}(x^{k+1}, y^{k+1}, z^{k+1}, \lambda^{k+1}) \\ & \stackrel{(1)}{\leq} \mathcal{L}_{r,M}(x^{k+1}, y^{k+1}, z^k, \lambda^{k+1}) \\ & = \mathcal{L}_{r,M}(x^{k+1}, y^{k+1}, z^k, \lambda^k) + \\ & \quad (\lambda^{k+1} - \lambda^k)^T (Ax^{k+1} - y^{k+1}) \\ & = \mathcal{L}_{r,M}(x^{k+1}, y^{k+1}, z^k, \lambda^k) + \frac{1}{\beta} \|\lambda^{k+1} - \lambda^k\|^2 \\ & \stackrel{(2)}{\leq} \mathcal{L}_{r,M}(x^{k+1}, y^{k+1}, z^k, \lambda^k) + \\ & \quad \frac{4(\beta_y^2 + (L_G^{r,M})^2)}{\beta} \|y^{k+1} - y^k\|^2 + \\ & \quad \frac{4\beta_y^2(1 + (L_G^{r,M} L_{zy}^{r^2, 3M/5, 4M/5})^2)}{\beta} \|y^k - y^{k-1}\|^2 \\ & \stackrel{(3)}{\leq} \mathcal{L}_{r,M}(x^k, y^k, z^k, \lambda^k) + \\ & \quad \left[\frac{4(\beta_y^2 + (L_G^{r,M})^2)}{\beta} - \frac{\beta_y}{2} \right] \|y^{k+1} - y^k\|^2 + \\ & \quad \frac{4\beta_y^2(1 + (L_G^{r,M} L_{zy}^{r^2, 3M/5, 4M/5})^2)}{\beta} \|y^k - y^{k-1}\|^2 - \\ & \quad \frac{\beta_x}{2} \|x^{k+1} - x^k\|^2, \end{aligned}$$

where, specifically, (1) is due to the definition of CLIP, (2) is due to Lemma 15, and (3) is due to Equation 41. The result follows by our choice of parameter:

$$\begin{aligned} & \frac{4(\beta_y^2 + (L_G^{r,M})^2)}{\beta} - \frac{\beta_y}{2} \\ & \leq - \frac{4\beta_y^2(1 + (L_G^{r,M} L_{zy}^{r^2, 3M/5, 4M/5})^2)}{\beta} - \frac{\beta_y}{4} \\ \Rightarrow \beta & \geq \frac{32\beta_y^2 + 16\beta_y^2(L_G^{r,M} L_{zy}^{r^2, 3M/5, 4M/5})^2 + 16(L_G^{r,M})^2}{\beta_y}, \end{aligned}$$

and all is proved.

We can further bound our proposed Lyapunov candidate $\Psi_{r,M}$ from below as follows:

Lemma 17. Taking Assumption 9 and Equation 32l, then we have for FT-ADMM_G:

$$\begin{aligned} & \Psi_{r,M}(x^k, y^k, z^k, \lambda^k, y^{k-1}) \\ & \geq f(x^k) + G_{r,M}(Ax^k, z^k) + \\ & \quad \left[\frac{\beta - L_G^{r,M}}{2} - \frac{\beta(\beta_y + L_G^{r,M} L_{zy}^{r^2, 3M/5, 4M/5})^2}{16\beta_y^2(1 + (L_G^{r,M} L_{zy}^{r^2, 3M/5, 4M/5})^2)} \right] \\ & \quad \|Ax^k - y^k\|^2 \\ & \geq f(x^k) + G_{r,M}(Ax^k, z^k) \geq \underline{G}_{r,M}. \end{aligned}$$

Proof. Our desired result then follows due to the following inequality:

$$\begin{aligned} & \mathcal{L}_{r,M}(x^k, y^k, z^k, \lambda^k) \\ & = f(x^k) + G_{r,M}(y^k, z^k) + \\ & \quad \frac{\beta}{2} \|Ax^k - y^k\|^2 + (\lambda^k)^T (Ax^k - y^k) \\ & \stackrel{(1)}{=} f(x^k) + G_{r,M}(y^k, z^k) + \frac{\beta}{2} \|Ax^k - y^k\|^2 + \\ & \quad (\beta_y(y^k - y^{k-1}) + \nabla_y G_{r,M}(y^k, z^{k-1}))^T (Ax^k - y^k) \\ & = f(x^k) + G_{r,M}(y^k, z^k) + \frac{\beta}{2} \|Ax^k - y^k\|^2 + \\ & \quad (\beta_y(y^k - y^{k-1}) + \nabla_y G_{r,M}(y^k, z^k))^T (Ax^k - y^k) + \\ & \quad (\nabla_y G_{r,M}(y^k, z^{k-1}) - \nabla_y G_{r,M}(y^k, z^k))^T (Ax^k - y^k) \\ & \stackrel{(2)}{\geq} f(x^k) + G_{r,M}(Ax^k, z^k) + \\ & \quad \frac{\beta - L_G^{r,M}}{2} \|Ax^k - y^k\|^2 + \beta_y(y^k - y^{k-1})^T (Ax^k - y^k) + \\ & \quad (\nabla_y G_{r,M}(y^k, z^{k-1}) - \nabla_y G_{r,M}(y^k, z^k))^T (Ax^k - y^k) \\ & \stackrel{(3)}{\geq} f(x^k) + G_{r,M}(Ax^k, z^k) + \frac{\beta - L_G^{r,M}}{2} \|Ax^k - y^k\|^2 - \\ & \quad \beta_y \|y^k - y^{k-1}\| \|Ax^k - y^k\| - \\ & \quad L_G^{r,M} \|z^k - z^{k-1}\| \|Ax^k - y^k\| \\ & \stackrel{(4)}{\geq} f(x^k) + G_{r,M}(Ax^k, z^k) + \frac{\beta - L_G^{r,M}}{2} \|Ax^k - y^k\|^2 \\ & \quad - (\beta_y + L_G^{r,M} L_{zy}^{r^2, 3M/5, 4M/5}) \|y^k - y^{k-1}\| \|Ax^k - y^k\| \\ & \geq f(x^k) + G_{r,M}(Ax^k, z^k) + \\ & \quad \left[\frac{\beta - L_G^{r,M}}{2} - \frac{\beta(\beta_y + L_G^{r,M} L_{zy}^{r^2, 3M/5, 4M/5})^2}{16\beta_y^2(1 + (L_G^{r,M} L_{zy}^{r^2, 3M/5, 4M/5})^2)} \right] \\ & \quad \|Ax^k - y^k\|^2 \\ & \quad - \frac{4\beta_y^2(1 + (L_G^{r,M} L_{zy}^{r^2, 3M/5, 4M/5})^2)}{\beta} \|y^k - y^{k-1}\|^2, \end{aligned}$$

and the result follows by our choice of Equation 32l. Here (1) is due to Equation 37, (2) and (3) are due to Lemma 13 iii) as well as Equation 30 and Equation 31, and (4) is due to Equation 38.

Connection between BC-ADMM and BC-ADMM_G

We show that the x^k, y^k, z^k sequences generated by BC-ADMM and BC-ADMM_G are the same under the proper parameter choice. as a result, BC-ADMM has the same convergence speed as BC-ADMM_G, aka. $O(\epsilon^{-2})$. To begin with, we need to analyze the behavior of the first iteration of BC-ADMM_G as follows:

Lemma 18. We take Assumption 9 and Assumption 14. The first iteration of BC-ADMM_G satisfies:

$$\begin{cases} \|x^1 - x^0\| \leq \frac{\delta}{2\|A\|} & (42a) \\ \|y^1 - Ax^1\| \leq \frac{\beta_y + L_G^{r,M}}{\beta_y + \beta - L_G^{r,M}} \frac{\delta}{2} \leq \frac{\delta}{2} & (42b) \\ \|y^1 - y^0\| \leq \delta & (42c) \\ z^1 = z(y^1). & (42d) \end{cases}$$

Proof. From our choice of initial value and Equation 37, we have:

$$\begin{aligned} 0 &= \beta_y(y^1 - Ax^0) + \nabla_y G_{r,M}(y^1, z^0) + \\ &\quad \beta(y^1 - Ax^1) - \nabla_y G_{r,M}(Ax^0, z^0) \\ &= \beta_y(y^1 - Ax^1 + Ax^1 - Ax^0) + \beta(y^1 - Ax^1) + \\ &\quad [\nabla_y G_{r,M}(y^1, z^0) - \nabla_y G_{r,M}(Ax^1, z^0)] + \\ &\quad [\nabla_y G_{r,M}(Ax^1, z^0) - \nabla_y G_{r,M}(Ax^0, z^0)]. \end{aligned}$$

By the triangle inequality, we have:

$$\begin{aligned} &(\beta_y + \beta)\|y^1 - Ax^1\| \\ &\leq (\beta_y + L_G^{r,M})\|A(x^1 - x^0)\| + L_G^{r,M}\|y^1 - Ax^1\| \\ \Rightarrow \|y^1 - Ax^1\| &\leq \frac{(\beta_y + L_G^{r,M})\|A\|}{\beta_y + \beta - L_G^{r,M}}\|x^1 - x^0\|. \end{aligned} \quad (43)$$

From our choice of initial value and Equation 35, we have:

$$\begin{aligned} 0 &= \beta_x(x^1 - x^0) + \nabla f(x^1) + \\ &\quad \beta A^T(Ax^1 - y^0) + A^T \nabla_y G_{r,M}(Ax^0, z^0) \\ &= \beta_x(x^1 - x^0) + [\nabla f(x^1) - \nabla f(x^0)] + \\ &\quad \beta A^T(Ax^1 - Ax^0) + \nabla f(x^0) + A^T \nabla_y G_{r,M}(Ax^0, z^0). \end{aligned}$$

Again by the triangle inequality, we have:

$$\begin{aligned} \beta_x\|x^1 - x^0\| &\leq \|[\beta_x I + \beta A^T A](x^1 - x^0)\| \\ &\leq L_f\|x^1 - x^0\| + \|\nabla f(x^0) + A^T \nabla_y G_{r,M}(Ax^0, z^0)\| \\ \Rightarrow \|x^1 - x^0\| &\leq \frac{1}{\beta_x - L_f} \|\nabla f(x^0) + A^T \nabla_y G_{r,M}(Ax^0, z^0)\| \\ &\leq \frac{1}{\beta_x - L_f} \|\nabla f(x^0) + A^T \nabla_y g(Ax^0, z^0)\| \stackrel{(1)}{\leq} \frac{\delta}{2\|A\|}, \end{aligned} \quad (44)$$

where in (1) we have used Equation 32e and we have proved Equation 42a. By combining Equation 43 and Equation 44, we have:

$$\|y^1 - Ax^1\| \leq \frac{\beta_y + L_G^{r,M}}{\beta_y + \beta - L_G^{r,M}} \frac{\delta}{2} \stackrel{(1)}{\leq} \frac{\delta}{2}, \quad (45)$$

where in (1) we have used Equation 32h and we have proved Equation 42b. Further, we have by combining Equation 43 and Equation 45:

$$\|y^1 - y^0\| \leq \|y^1 - Ax^1\| + \|Ax^1 - Ax^0\| \leq \delta, \quad (46)$$

and we have proved Equation 42c. Finally, we have the following chain of inequality:

$$\begin{aligned} G_{r,M}(y^1, z^1) &\stackrel{(4)}{=} G_{r,M}(y^1, z(y^1)) \stackrel{(3)}{=} g(y^1, z(y^1)) \\ &\stackrel{(2)}{\leq} g(y^1, z(y^0)) \stackrel{(1)}{=} G_{r,M}(y^1, z(y^0)) = G_{r,M}(y^1, z^0). \end{aligned} \quad (47)$$

To see the above inequality, we note that $(y^0, z^0) \in \mathcal{S}_{\mathcal{Z}}(r^1, 2M/5) \subseteq \mathcal{S}_{\mathcal{Z}}(r^4, 4M/5)$ and $\|y^1 - y^0\| \leq \delta \leq \delta^{r^4, 4M/5, M}$ by Equation 46. Now by Lemma 13, we have (1) and (2) is due to optimality of $z(y^1)$. To see (3), we can apply Lemma 12 to show that $\|z(y^1) - z(y^0)\| \leq L_{zy}^{r^2, 3M/5, 4M/5} \|y^1 - y^0\|$ so that we have:

$$\begin{aligned} \|(y^1, z(y^1))\| &\leq \|(y^0, z^0)\| + \|(y^1, z(y^1)) - (y^0, z^0)\| \\ &\leq r^1 + \delta(1 + L_{zy}^{r^2, 3M/5, 4M/5}) \leq r. \end{aligned}$$

Also note that we have already shown by (1) and (2) that $g(y^1, z(y^1)) \leq M$, so we have $(y^1, z(y^1)) \in \mathcal{S}(r, M)$ where $G_{r,M} = g$, which is (3). Now by Equation 34d and the definition of CLIP, we have proved (4), which is exactly Equation 42d and all is proved.

Next, we show that the Lyapunov candidate in the first iteration can be upper bounded:

Lemma 19. We take Assumption 9 and Assumption 14. The Lyapunov candidate in first iteration of BC-ADMM_G satisfies $\Psi_{r,M}(x^1, y^1, z^1, \lambda^1, y^0) \leq 2M/5$.

Proof. This is due to the following inequality:

$$\begin{aligned}
& f(x^0) + g(Ax^0, z^0) \\
& \stackrel{(1)}{=} f(x^0) + G_{r,M}(Ax^0, z^0) = \mathcal{L}_{r,M}(x^0, y^0, z^0, \lambda^0) \\
& \stackrel{(2)}{\geq} \mathcal{L}_{r,M}(x^1, y^1, z^0, \lambda^0) + \frac{\beta_x}{2} \|x^1 - x^0\|^2 + \frac{\beta_y}{2} \|y^1 - y^0\|^2 \\
& \stackrel{(3)}{\geq} \mathcal{L}_{r,M}(x^1, y^1, z^1, \lambda^0) + \frac{\beta_x}{2} \|x^1 - x^0\|^2 + \frac{\beta_y}{2} \|y^1 - y^0\|^2 \\
& = \mathcal{L}_{r,M}(x^1, y^1, z^1, \lambda^1) - (\lambda^1 - \lambda^0)^T (Ax^1 - y^1) + \\
& \quad \frac{\beta_x}{2} \|x^1 - x^0\|^2 + \frac{\beta_y}{2} \|y^1 - y^0\|^2 \\
& = \mathcal{L}_{r,M}(x^1, y^1, z^1, \lambda^1) - \beta \|Ax^1 - y^1\|^2 + \\
& \quad \frac{\beta_x}{2} \|x^1 - x^0\|^2 + \frac{\beta_y}{2} \|y^1 - y^0\|^2 \\
& \geq \Psi_{r,M}(x^1, y^1, z^1, \lambda^1, y^0) - \beta \|Ax^1 - y^1\|^2 + \\
& \quad \frac{\beta_x}{2} \|x^1 - x^0\|^2 + \frac{\beta_y}{2} \|y^1 - y^0\|^2 - \\
& \quad \frac{4\beta_y^2(1 + (L_{zy}^{r,M} L_{zy}^{r^2, 3M/5, 4M/5})^2)}{\beta} \|y^1 - y^0\|^2 \\
& \stackrel{(4)}{\geq} \Psi_{r,M}(x^1, y^1, z^1, \lambda^1, y^0) - \beta \|Ax^1 - y^1\|^2 + \\
& \quad \frac{\beta_x}{2} \|x^1 - x^0\|^2 + \frac{\beta_y}{4} \|y^1 - y^0\|^2 \\
& \stackrel{(5)}{\geq} \Psi_{r,M}(x^1, y^1, z^1, \lambda^1, y^0) - \frac{\beta(\beta_y + L_G^{r,M})^2}{(\beta_y + \beta - L_G^{r,M})^2} \frac{\delta^2}{4} \\
& \stackrel{(6)}{\geq} \Psi_{r,M}(x^1, y^1, z^1, \lambda^1, y^0) - \frac{\delta^2}{\beta}.
\end{aligned}$$

Here (1) uses our choice of initial guess and the fact that $(Ax^0, z^0) \in \mathcal{S}(r, M)$. (2) is due to Equation 41. (3) is due to Equation 47. (4) is due to Equation 32i. (5) is due to Equation 42b. (6) is due to Equation 32j. Finally, our desired result is due to Equation 32k. The proof is complete.

We have established all the results about the initial iteration and we are now ready to show the equivalence between BC-ADMM and BC-ADMM_G. We prove the following result by induction:

Lemma 20. We take Assumption 9 and Assumption 14. Then every iteration generated by BC-ADMM_G satisfies:

$$\begin{cases}
y^k \text{ by BC-ADMM and BC-ADMM}_G \text{ is same} & (48a) \\
z^k = z(y^k) & (48b) \\
f(x^k) + g(Ax^k, z^k) \leq 2M/5. & (48c)
\end{cases}$$

In summary, the x^k, y^k, z^k sequence generated by BC-ADMM_G and BC-ADMM are the same.

Proof. We use mathematical induction. The case with $k = 0$ is trivially due to our choice of initial value. Base Step ($k = 1$, Equation 48b and Equation 48c): We have $f(x^0) + g(Ax^0, z^0) \leq M/5 \leq 2M/5$ by our initial guess so $(Ax^0, z^0) \in \mathcal{S}_Z(r^1, 2M/5)$. We have $\|Ax^0 - Ax^1\| \leq \delta \leq \delta^{r^1, 2M/5, 3M/5}$ by Equation 42a and:

$$\begin{aligned}
\|(Ax^1, z^0)\| & \leq \|(Ax^0, z^0)\| + \|Ax^1 - Ax^0\| \\
& \leq r^1 + \delta^{r^1, 2M/5, 3M/5} = r^2,
\end{aligned}$$

leading to $(Ax^1, z^0) \in \mathcal{S}_Z(r^2, 3M/5)$. We also have $\|Ax^0 - y^0\| = 0 \leq \delta^{r^1, 2M/5, 3M/5}$ so $(y^0, z^0) \in \mathcal{S}_Z(r^1, 3M/5) \subseteq \mathcal{S}_Z(r^2, 3M/5)$. Next, we have $\|Ax^1 - y^1\| \leq \delta \leq \delta^{r^2, 3M/5, 4M/5}$ by Equation 42b and:

$$\begin{aligned}
\|(y^1, z^0)\| & \leq \|(Ax^1, z^0)\| + \|y^1 - Ax^1\| \\
& \leq r^2 + \delta^{r^2, 3M/5, 4M/5} = r^3,
\end{aligned}$$

leading to $(y^1, z^0) \in \mathcal{S}_Z(r^3, 4M/5)$. Then, we have $(y^0, z^0) \in \mathcal{S}_Z(r^2, 3M/5)$ and $\|y^1 - y^0\| \leq \delta \leq \delta^{r^2, 3M/5, 4M/5}$ so by Lemma 12 and Equation 42d:

$$\|z^1 - z^0\| = \|z(y^1) - z(y^0)\| \leq L_{zy}^{r^2, 3M/5, 4M/5} \|y^1 - y^0\|.$$

Now by optimality of $z(y^1)$, we know that $g(y^1, z^1) = g(y^1, z(y^1)) \leq g(y^1, z^0) \leq 4M/5$ and:

$$\begin{aligned}
\|(y^1, z^1)\| & \leq \|(y^1, z^0)\| + \|z^1 - z^0\| \\
& \leq r^3 + L_{zy}^{r^2, 3M/5, 4M/5} \delta = r^4,
\end{aligned}$$

leading to $(y^1, z^1) \in \mathcal{S}_Z(r^4, 4M/5)$. Finally, we have $\|Ax^1 - y^1\| \leq \delta \leq \delta^{r^4, 4M/5, M}$ so that: $(Ax^1, z^1) \in \mathcal{S}_Z(r^4 + \delta^{r^4, 4M/5, M}, M) = \mathcal{S}_Z(r, M)$. However, we also have:

$$\begin{aligned}
0 & \leq f(x^1) + g(Ax^1, z^1) = f(x^1) + G_{r,M}(Ax^1, z^1) \\
& \stackrel{(2)}{\leq} \Psi_{r,M}(x^1, y^1, z^1, \lambda^1, y^0) \stackrel{(1)}{\leq} 2M/5,
\end{aligned}$$

where (1) is due to Lemma 19 and (2) is due to Lemma 17. We have essentially proved Equation 48c and Equation 48b is due to Equation 42d. Now we have finished proving the base case.

Inductive Step ($k > 1$, Equation 48b and Equation 48c): We have $f(x^k) + g(Ax^k, z^k) \leq M/5 \leq 2M/5$ by our inductive assumption so $(Ax^k, z^k) \in \mathcal{S}_Z(r^1, 2M/5)$. We have the following estimate in the change of x^k :

$$\begin{aligned}
& \|x^{k+1} - x^k\| \\
& \stackrel{(1)}{\leq} \sqrt{\frac{4 \left(\Psi_{r,M}(x^k, y^k, z^k, \lambda^k, y^{k-1}) - \Psi_{r,M}(x^{k+1}, y^{k+1}, z^{k+1}, \lambda^{k+1}, y^k) \right)}{\beta_x}} \\
& \stackrel{(2)}{\leq} \sqrt{\frac{4(\Psi_{r,M}(x^1, y^1, z^1, \lambda^1, y^0) - \underline{G}_{r,M})}{\beta_x}} \\
& \stackrel{(3)}{\leq} \sqrt{\frac{4(2M/5 - \underline{G}_{r,M})}{\beta_x}} \stackrel{(4)}{\leq} \frac{\delta}{\|A\|},
\end{aligned}$$

where (1) is due to Equation 40, (2) is due to Lemma 17, (3) is due to Lemma 19, and (4) is due to Equation 32f. We further have the following estimate in the change of y^k :

$$\begin{aligned}
& \|y^{k+1} - y^k\| \\
& \stackrel{(1)}{\leq} \sqrt{\frac{4 \left(\Psi_{r,M}(x^k, y^k, z^k, \lambda^k, y^{k-1}) - \Psi_{r,M}(x^{k+1}, y^{k+1}, z^{k+1}, \lambda^{k+1}, y^k) \right)}{\beta_y}} \\
& \stackrel{(2)}{\leq} \sqrt{\frac{4(\Psi_{r,M}(x^1, y^1, z^1, \lambda^1, y^0) - \underline{G}_{r,M})}{\beta_y}} \\
& \stackrel{(3)}{\leq} \sqrt{\frac{4(2M/5 - \underline{G}_{r,M})}{\beta_y}} \stackrel{(4)}{\leq} \delta,
\end{aligned}$$

where (1) is due to Equation 40, (2) is due to Lemma 17, (3) is due to Lemma 19, and (4) is due to Equation 32g. We thus have $\|Ax^k - Ax^{k+1}\| \leq \delta \leq \delta^{r^1, 2M/5, 3M/5}$ and:

$$\begin{aligned} \|(Ax^{k+1}, z^k)\| &\leq \|(Ax^k, z^k)\| + \|Ax^{k+1} - Ax^k\| \\ &\leq r^1 + \delta^{r^1, 2M/5, 3M/5} = r^2, \end{aligned}$$

leading to $(Ax^{k+1}, z^k) \in \mathcal{S}_{\mathcal{Z}}(r^2, 3M/5)$. We further have the following bound on $\|Ax^{k+1} - y^{k+1}\|$ or $\|Ax^k - y^k\|$:

$$\begin{aligned} &\max(\|Ax^{k+1} - y^{k+1}\|^2, \|Ax^k - y^k\|^2) \\ &\stackrel{(1)}{\leq} \sqrt{\frac{2M/5 - \underline{G}_{r,M}}{\left[\frac{\beta - L_{\underline{G}}^{r,M}}{2} - \frac{\beta(\beta_y + L_{\underline{G}}^{r,M} L_{zy}^{r^2, 3M/5, 4M/5})^2}{16\beta_y^2(1 + (L_{\underline{G}}^{r,M} L_{zy}^{r^2, 3M/5, 4M/5})^2)}\right]}} \stackrel{(2)}{\leq} \delta, \end{aligned}$$

where (1) is due to Lemma 17 and Lemma 19, while (2) is due to Equation 32l. Combined with the fact that $(Ax^k, z^k) \in \mathcal{S}_{\mathcal{Z}}(r^1, 2M/5)$, we know that $(y^k, z^k) \in \mathcal{S}(r^2, 3M/5)$. Next, we combine the fact that $(Ax^{k+1}, z^k) \in \mathcal{S}_{\mathcal{Z}}(r^2, 3M/5)$ and:

$$\begin{aligned} \|(y^{k+1}, z^k)\| &\leq \|(Ax^{k+1}, z^k)\| + \|y^{k+1} - Ax^{k+1}\| \\ &\leq r^2 + \delta^{r^2, 3M/5, 4M/5} = r^3, \end{aligned}$$

leading to $(y^{k+1}, z^k) \in \mathcal{S}_{\mathcal{Z}}(r^3, 4M/5)$. Then, we have $(y^k, z^k) \in \mathcal{S}_{\mathcal{Z}}(r^2, 3M/5)$ and $\|y^{k+1} - y^k\| \leq \delta \leq \delta^{r^2, 3M/5, 4M/5}$ so by Lemma 12 and:

$$\begin{aligned} \|z^{k+1} - z^k\| &= \|z(y^{k+1}) - z(y^k)\| \\ &\leq L_{zy}^{r^2, 3M/5, 4M/5} \|y^{k+1} - y^k\|, \end{aligned}$$

and Equation 48b follows by an identical argument as in Equation 47. Now optimality of $z(y^{k+1})$, we know that $g(y^{k+1}, z^{k+1}) = g(y^{k+1}, z(y^{k+1})) \leq g(y^{k+1}, z^k) \leq 4M/5$ as well as:

$$\begin{aligned} \|(y^{k+1}, z^{k+1})\| &\leq \|(y^{k+1}, z^k)\| + \|z^{k+1} - z^k\| \\ &\leq r^3 + L_{zy}^{r^2, 3M/5, 4M/5} \delta = r^4, \end{aligned}$$

leading to $(y^{k+1}, z^{k+1}) \in \mathcal{S}_{\mathcal{Z}}(r^4, 4M/5)$. Finally, we have $\|Ax^{k+1} - y^{k+1}\| \leq \delta \leq \delta^{r^4, 4M/5, M}$ so that: $(Ax^{k+1}, z^{k+1}) \in \mathcal{S}_{\mathcal{Z}}(r^4 + \delta^{r^4, 4M/5, M}, M) = \mathcal{S}_{\mathcal{Z}}(r, M)$. However, we also have:

$$\begin{aligned} 0 &\leq f(x^{k+1}) + g(Ax^{k+1}, z^{k+1}) \\ &= f(x^{k+1}) + G_{r,M}(Ax^{k+1}, z^{k+1}) \\ &\stackrel{(2)}{\leq} \Psi_{r,M}(x^{k+1}, y^{k+1}, z^{k+1}, \lambda^{k+1}, y^k) \stackrel{(1)}{\leq} 2M/5, \end{aligned}$$

where (1) is due to Lemma 19 and (2) is due to Lemma 17. We have essentially proved Equation 48c. Now we have finished proving the inductive case.

Base & Inductive Step ($k \geq 1$, Equation 48a): From the analysis above, we know that $(y^k, z^k) \in \mathcal{S}_{\mathcal{Z}}(r^2, 3M/5)$ for every $k \geq 1$ and $\|y^{k+1} - y^k\| \leq \delta \leq \delta^{r^2, 3M/5, 4M/5}$. As a result, $(y^{k+1}, z^k) \in \mathcal{S}(r^3, 4M/5)$. Therefore, the gradient of $G_{r,M}$ and g matches and we have at y^{k+1} generated by BC-ADMM_G:

$$\begin{aligned} 0 &= \nabla_y \left[\mathcal{L}_{r,M}(x^{k+1}, y^{k+1}, z^k, \lambda^k) + \frac{\beta_y}{2} \|y^{k+1} - y^k\|^2 \right] \\ &= \nabla_y \left[\mathcal{L}(x^{k+1}, y^{k+1}, z^k, \lambda^k) + \frac{\beta_y}{2} \|y^{k+1} - y^k\|^2 \right]. \end{aligned}$$

But since the objective function in Equation 13c of BC-ADMM is strongly convex, whose optimal solution is unique, we must have y^{k+1} generated by BC-ADMM is equivalent to that generated by BC-ADMM_G.

Finally, we still need to show that BC-ADMM_G and BC-ADMM generates the same sequence. Note that the two algorithm uses different steps in the update of y^k and z^k . But we have shown that $z^k = z(y^k)$ for every k and that the update from the same y^k leads to the same y^{k+1} . Our desired result follows.

As our last step of analysis, we establish the convergence speed of BC-ADMM_G in the following result:

Lemma 21. *Under Assumption 9 and Assumption 14, BC-ADMM_G converges at a speed of $O(\epsilon^{-2})$ to ϵ -stationary point of Equation 11.*

Proof. We can repeatedly applying Lemma 16 and, by letting $\beta_{xy} = \min(\beta_x, \beta_y)$, we have:

$$\begin{aligned} &\frac{\beta_{xy}}{4} \sum_{k=1}^K \|x^{k+1} - x^k\|^2 + \|y^{k+1} - y^k\|^2 \\ &\leq \frac{\beta_x}{4} \sum_{k=1}^K \|x^{k+1} - x^k\|^2 + \frac{\beta_y}{4} \sum_{k=1}^K \|y^{k+1} - y^k\|^2 \\ &\leq \Psi_{r,M}(x^1, y^1, \lambda^1, y^0) - \Psi_{r,M}(x^{K+1}, y^{K+1}, \lambda^{K+1}, y^K). \end{aligned}$$

Adding above equations twice and define $\theta^k = \|x^{k+1} - x^k\|^2 + \|y^{k+1} - y^k\|^2 + \|x^k - x^{k-1}\|^2 + \|y^k - y^{k-1}\|^2$, we have by the second term in parameter choice and Lemma 17:

$$\begin{aligned} &\frac{\beta_{xy}}{4} \sum_{k=2}^K \theta^k \\ &= \frac{\beta_{xy}}{4} \left[\sum_{k=1}^{K-1} \|x^{k+1} - x^k\|^2 + \|y^{k+1} - y^k\|^2 + \sum_{k=2}^K \|x^{k+1} - x^k\|^2 + \|y^{k+1} - y^k\|^2 \right] \\ &\leq 2\Psi_{r,M}(x^1, y^1, \lambda^1, y^0) - 2\Psi_{r,M}(x^{K+1}, y^{K+1}, \lambda^{K+1}, y^K) \\ &\leq 2(\Psi_{r,M}(x^1, y^1, \lambda^1, y^0) - \underline{G}_{r,M}). \end{aligned}$$

Therefore, we can bound the change in both x and y over consecutive iterations as follows:

$$\min_{k=2, \dots, K} \theta^k \leq \frac{8(\Psi_{r,M}(x^1, y^1, \lambda^1, y^0) - \underline{G}_{r,M})}{\beta_{xy} K}. \quad (49)$$

Now define $k = \operatorname{argmin}_{k=2, \dots, K} \theta^k$ and we bound the norm of objective's gradient with respect to x^k .

$$\begin{aligned}
& \|\nabla f(x^k) + A^T \nabla_y g(Ax^k, z^k)\|^2 \\
&= \|\nabla f(x^k) + A^T \nabla_y G_{r,M}(Ax^k, z^k)\|^2 \\
&= \|\nabla f(x^k) + A^T \lambda^k + A^T (\nabla_y G_{r,M}(y^k, z^{k-1}) - \lambda^k) + \\
&\quad A^T (\nabla_y G_{r,M}(y^k, z^k) - \nabla_y G_{r,M}(y^k, z^{k-1})) + \\
&\quad A^T (\nabla_y G_{r,M}(Ax^k, z^k) - \nabla_y G_{r,M}(y^k, z^k))\|^2 \\
&\stackrel{(1)}{=} \|\beta_x(x^k - x^{k-1}) - \beta A^T(y^k - y^{k-1}) - \beta_y A^T(y^k - y^{k-1}) + \\
&\quad A^T (\nabla_y G_{r,M}(y^k, z^k) - \nabla_y G_{r,M}(y^k, z^{k-1})) + \\
&\quad A^T (\nabla_y G_{r,M}(Ax^k, z^k) - \nabla_y G_{r,M}(y^k, z^k))\|^2 \\
&\stackrel{(2)}{\leq} 4\beta_x^2 \|x^k - x^{k-1}\|^2 + 4(\beta + \beta_y)^2 \|A\|^2 \|y^k - y^{k-1}\|^2 + \\
&\quad 4(L_G^{r,M})^2 \|A\|^2 [\|z^k - z^{k-1}\|^2 + \|Ax^k - y^k\|^2] \\
&= 4\beta_x^2 \|x^k - x^{k-1}\|^2 + 4(\beta + \beta_y)^2 \|A\|^2 \|y^k - y^{k-1}\|^2 + \\
&\quad 4(L_G^{r,M})^2 \|A\|^2 \left[\|z^k - z^{k-1}\|^2 + \frac{\|\lambda^k - \lambda^{k-1}\|^2}{\beta^2} \right] \\
&\stackrel{(3)}{\leq} 4\beta_x^2 \|x^k - x^{k-1}\|^2 + 4(\beta + \beta_y)^2 \|A\|^2 \|y^k - y^{k-1}\|^2 + \\
&\quad 4(L_G^{r,M} L_{zy}^{r^2, 3M/5, 4M/5})^2 \|A\|^2 \|y^k - y^{k-1}\|^2 + 4 \frac{(L_G^{r,M})^2}{\beta^2} \|A\|^2 \\
&\quad \left[4(\beta_y^2 + (L_G^{r,M})^2) \|y^{k+1} - y^k\|^2 + \right. \\
&\quad \left. 4\beta_y^2 (1 + (L_G^{r,M} L_{zy}^{r^2, 3M/5, 4M/5})^2) \|y^k - y^{k-1}\|^2 \right] \\
&= O(\theta^k).
\end{aligned}$$

Here (1) is due to Equation 36 and Equation 37, (2) is due to Lemma 13 iii), and (3) is due to Lemma 15. Combined, we have:

$$\begin{aligned}
& \|\nabla f(x^k) + A^T \nabla_y g(Ax^k, z^k)\|^2 \\
&= O\left(\frac{8(\Psi_{r,M}(x^1, y^1, \lambda^1, y^0) - \underline{G}_{r,M})}{\beta_{xy} K}\right) = O(\epsilon^2). \quad (50)
\end{aligned}$$

Next, we bound the norm of objective's gradient with respect to z^k . By (Rockafellar and Wets 2009, Theorem 8.15) and the fact that $z^k = z(y^k)$ due to Lemma 20, we have $-\nabla_z g(y^k, z^k) = N_{\mathcal{Z}}(z^k)$, with $N_{\mathcal{Z}}(z^k)$ being the normal cone of $\mathcal{Z}$ at z^k . As a result, we have:

$$\begin{aligned}
& \operatorname{dist}(N_{\mathcal{Z}}(z^k), -\nabla_z g(Ax^k, z^k))^2 \\
&= \operatorname{dist}(N_{\mathcal{Z}}(z^k), -\nabla_z G_{r,M}(Ax^k, z^k))^2 \\
&\stackrel{(1)}{\leq} \|\nabla_z G_{r,M}(Ax^k, z^k) - \nabla_z G_{r,M}(y^k, z^k)\|^2 \\
&\stackrel{(2)}{=} (L_G^{r,M})^2 \|Ax^k - y^k\|^2 = \frac{(L_G^{r,M})^2}{\beta^2} \|\lambda^k - \lambda^{k-1}\|^2 \\
&\stackrel{(3)}{\leq} \frac{(L_G^{r,M})^2}{\beta^2} \left[4(\beta_y^2 + (L_G^{r,M})^2) \|y^{k+1} - y^k\|^2 + \right. \\
&\quad \left. 4\beta_y^2 (1 + (L_G^{r,M} L_{zy}^{r^2, 3M/5, 4M/5})^2) \|y^k - y^{k-1}\|^2 \right] \\
&= O(\theta^k).
\end{aligned}$$

Here (1) is due to the fact that $-\nabla_z G_{r,M}(y^k, z^k) = -\nabla_z g(y^k, z^k) \in N_{\mathcal{Z}}(z^k)$ and that $\operatorname{dist}(N_{\mathcal{Z}}(z^k), a) \leq \|a - b\| + \operatorname{dist}(N_{\mathcal{Z}}(z^k), b)$, (2) is due to Lemma 13 iii), and (3) is due to Lemma 15. Combined we have:

$$\begin{aligned}
& \operatorname{dist}(N_{\mathcal{Z}}(z^k), -\nabla_z g(Ax^k, z^k))^2 \\
&= O\left(\frac{8(\Psi_{r,M}(x^1, y^1, \lambda^1, y^0) - \underline{G}_{r,M})}{\beta_{xy} K}\right) = O(\epsilon^2). \quad (51)
\end{aligned}$$

Our desired result follows by combining Equation 50 and Equation 51.

Theorem 22. Taking Assumption 9 and under sufficiently large β_x, β_y, β , each iteration generated by BC-ADMM satisfies $f(x^k) + g(Ax^k, z^k) < \infty$. Further, BC-ADMM converges to the ϵ -stationary solution of Equation 10 with an oracle complexity of $O(\epsilon^{-2})$.

Proof. Combine the result of Lemma 20 and Lemma 21.

Convergence of Algorithm 1

In Algorithm 1, we introduced several properties to allow automatic tuning of parameters and progressively introduce constraints. To formally argue for the convergence, we use the observation that with z fixed, our objective $f(\bullet) + g(A\bullet, z)$ is convex and Algorithm 1 becomes a special, two-block case of the proximal ADMM algorithm analyzed in Deng et al. (2017). According to their analysis, the parameters β, β_x, β_y need to satisfy the following conditions:

$$\begin{cases} 1 > \epsilon_x + \epsilon_y \\ \beta_x I \succ \beta(1/\epsilon_x - 1)A^T A \\ \beta_y > \beta(1/\epsilon_y - 1) \end{cases} \quad (52)$$

To prove convergence, we show that our parameter choices in Algorithm 1 always satisfies Equation 52 by construction. It also satisfies Assumption 14 after a sufficient number of parameter tuning iterations (below Line 34). As a result, the feasibility of our solution is ensured by Theorem 22. Our result is formally claimed in Theorem 6 and proved here.

Proof of Theorem 6. We note that Line 17 ensures that x^* stores the best solution with the smallest $f(x^*) + g(Ax^*, z^*) \leq f(x^0) + g(Ax^0, z^0) < \infty$. We prove several properties of Algorithm 1.

We first prove that the parameter tuning iteration below Line 34 will be entered only finitely many times. This is because the parameter tuning iteration would restart the simulation from the best solution at $\langle x, z \rangle^{k+1} \leftarrow \langle x, z \rangle^*$ while resetting y^{k+1} and λ^{k+1} . Further, each parameter tuning iteration would increase β, β_y indefinitely. Since $\kappa > \kappa_y$, β would be ultimately larger than β_y by an arbitrarily large factor. We could easily see that β_x would also be indefinitely large, which is obvious from the following rewrite:

$$\beta_x \leftarrow \beta(1/\gamma + \beta/\beta_y - 1) \|A^T A\|/\gamma.$$

As a result, all the requirements from Assumption 14 would be satisfied within finitely many parameter tuning iterations, after which all the iterations are strictly feasible by Theorem 22 and the condition below Line 34 will never be met. As a result, we can safely ignore the finitely many iterations and assume that β_x, β_y, β are never modified while all iterations are feasible.

Second, we investigate the number of times z^* will be updated to $z(y^{k+1})$. Note that we use a lazy strategy and update z^* only when the two conditions below Line 17 are met. Suppose these conditions are not met and z^* is not updated, then our algorithm becomes the

conventional ADMM Deng and Yin (2016) minimizing the convex objective function $f(\bullet) + g(A\bullet, z^*)$. Due to our parameter choice, Equation 52 is always satisfied and (Deng and Yin 2016, Theorem 2.3) applies, which shows that $f(x^{k+1}) + g(y^{k+1}, z^*)$ converges to the (globally) optimal solution of that objective function denoted as $x^\dagger$, i.e. $\lim_{k \rightarrow \infty} x^k = x^\dagger$ and $\lim_{k \rightarrow \infty} y^k = Ax^\dagger$. This also implies that $\lim_{k \rightarrow \infty} \Phi^{k+1} = 0$. As a result, the second condition below Line 17, i.e., $\Phi^{k+1} \leq \eta^K$, will be met for sufficiently large k . In order for z^* to be updated, we also need the first condition to be satisfied. To this end, we consider two cases. Case I: If $g(Ax^\dagger, z(Ax^\dagger)) < g(Ax^\dagger, z^*)$, then we have:

$$\begin{aligned} \lim_{k \rightarrow \infty} f(x^{k+1}) + g(Ax^{k+1}, z^{k+1}) &\stackrel{(1)}{=} f(x^\dagger) + g(Ax^\dagger, z(Ax^\dagger)) \\ &= f(x^\dagger) + g(Ax^\dagger, z^*) + (g(Ax^\dagger, z(Ax^\dagger)) - g(Ax^\dagger, z^*)) \\ &\stackrel{(2)}{\leq} f(x^*) + g(Ax^*, z^*) + (g(Ax^\dagger, z(Ax^\dagger)) - g(Ax^\dagger, z^*)). \end{aligned}$$

In (1) above, we used the fact that $\lim_{k \rightarrow \infty} z^{k+1} = \lim_{k \rightarrow \infty} z(y^{k+1}) = z(y^\dagger) = z(Ax^\dagger)$ because $z(y)$ is a locally Lipschitz function of y by our Assumption 9 vi). (2) is due to optimality of $x^\dagger$. We further have:

$$\begin{aligned} \lim_{k \rightarrow \infty} \Lambda^{k+1} &= f(x^*) + g(Ax^*, z^*) + \\ &\quad (1 - \eta)(g(Ax^\dagger, z(Ax^\dagger)) - g(Ax^\dagger, z^*)). \end{aligned}$$

Put together, we know that $\lim_{k \rightarrow \infty} f(x^{k+1}) + g(Ax^{k+1}, z^{k+1}) < \lim_{k \rightarrow \infty} \Lambda^{k+1}$ so that the first condition below Line 17 will be satisfied after finitely many iterations where z^* will be updated. Case II: If $g(Ax^\dagger, z(Ax^\dagger)) = g(Ax^\dagger, z^*)$, then $z^* = z(Ax^\dagger)$ due to the uniqueness of $z(Ax^\dagger)$ by our Assumption 9 vi) and z^* might never be updated further. Suppose this is the case, due to the fact that $\lim_{k \rightarrow \infty} \Phi^{k+1} = 0$, we can easily verify that (x^{k+1}, y^{k+1}, z^*) is the desired ϵ -stationary solution of Equation 10 for sufficiently large k .

As our final step, we consider the case where z^* is updated infinitely many times. In this case, we denote the subsequence of iterations that update z^k as $z^{k(i)}$ with $i = 1, 2, \dots$. In other words, we assume that, at every iteration $k(i)$, the two conditions below Line 17 are met and we have $z^* = z^{k(i)+1} = z(y^{k(i)+1})$. The sequence $\{(x, y, z)^{k(i)+1}\}$ is bounded so that there exists a convergent subsequence. Without a loss of generality in the following arguments, we can assume the entire sequence $\{(x, y, z)^{k(i)+1}\}$ converges to some $(x, y, z)^\dagger$. Clearly, we have $z^\dagger = z(y^\dagger)$ because $z(y)$ is a locally Lipschitz function of y by our Assumption 9 vi).

We first claim that $\lim_{i \rightarrow \infty} g(y^{k(i)+1}, z^{k(i)+1}) - g(y^{k(i)+1}, z^{k(i)}) = 0$. This is because the sequence $\{f(x^*) + g(Ax^*, z^*)\}$ is monotonically decreasing and it reduces by at least $(1 - \eta)(g(y^{k(i)+1}, z^{k(i)+1}) - g(y^{k(i)+1}, z^{k(i)}))$ at the $k(i)$ th iteration due to the first condition below Line 17. If $\liminf_{i \rightarrow \infty} g(y^{k(i)+1}, z^{k(i)+1}) - g(y^{k(i)+1}, z^{k(i)}) < 0$, then $f(x^*) + g(Ax^*, z^*)$ would tend to $-\infty$, which contradicts the fact that both f and g are positive functions. We then claim that $\lim_{i \rightarrow \infty} z^{k(i)+1} - z^{k(i)} = 0$. Indeed, suppose otherwise, there exists some $\epsilon > 0$ where $\|z^{k(i)+1} - z^{k(i)}\| > \epsilon$ for infinitely many i . We can assume that $z^{k(i)}$ converges (otherwise, choose a convergent subsequence), but with a limit different

from $z(y^\dagger)$. But we know by our previous claim that $\lim_{i \rightarrow \infty} g(y^{k(i)+1}, z^{k(i)+1}) - g(y^{k(i)+1}, z^{k(i)}) = g(y^\dagger, z(y^\dagger)) - \lim_{i \rightarrow \infty} g(y^\dagger, z^{k(i)}) = 0$. This contradicts our Assumption 9 vi) that $z(y^\dagger)$ is unique. Our second claim immediately implies that $\lim_{i \rightarrow \infty} z^{k(i)} = z(y^\dagger)$, which combined with the fact that $\lim_{k \rightarrow \infty} \Phi^{k(i)+1} = 0$ again leads to ϵ -stationary solution of Equation 10 for sufficiently large k .

Remark 23. In Theorem 6, we have assumed that the constraint detector below Line 19 has never been invoked. In practice, there are finitely many constraints so that the detector will be invoked for at most finitely many times. Further, a well-defined constraint detector only ignores constraint function terms when the value of these terms are smaller than some user-defined threshold. It is not difficult to show asymptotic convergence under this setting.

Special Cases of Theorem 22

In this section, we prove several special cases of Theorem 22, adapting it to different application domains. First, we show that two terms g_1 and g_2 can be summed together and still satisfy Assumption 9.

Proof of Corollary 8. Define $y = (y_1, y_2)$, $z = (z_1, z_2)$, and $z(y) = (z_1(y_1), z_2(y_2))$. Suppose $z_1(y_1)$ and $z_2(y_2)$ are unique global minimizers of $g_1(y_1, \bullet) + \iota_{\mathcal{Z}_1}(\bullet)$ and $g_2(y_2, \bullet) + \iota_{\mathcal{Z}_2}(\bullet)$, then obviously $z(y)$ is the unique global minimizer of $g(y, \bullet) + \iota_{\mathcal{Z}}(\bullet)$, proving the first part of Assumption 9 vi).

Now suppose $(y, z) \in \mathcal{S}_{\mathcal{Z}}(r, M)$, then clearly $(y_1, z_1) \in \mathcal{S}_{\mathcal{Z}_1}(r, M)$ and $(y_2, z_2) \in \mathcal{S}_{\mathcal{Z}_2}(r, M)$ and by our assumption we have $(y_1, z_1(y_1)) \in \mathcal{S}_{\mathcal{Z}_1}(\bar{r}_{z_1 y}^{r, M}, M)$ and $(y_2, z_2(y_2)) \in \mathcal{S}_{\mathcal{Z}_2}(\bar{r}_{z_2 y}^{r, M}, M)$. Combined, we have $\|(y, z)\| \leq \|(y_1, z_1)\| + \|(y_2, z_2)\| \leq \bar{r}_{z_1 y}^{r, M} + \bar{r}_{z_2 y}^{r, M} \triangleq \bar{r}_{zy}^{r, M}$. Therefore, we have $(y, z(y)) \in \mathcal{S}_{\mathcal{Z}}(\bar{r}_{zy}^{r, M}, M)$, proving the second part of Assumption 9 vi).

Finally, suppose $(y, z(y)) \in \mathcal{S}_{\mathcal{Z}}(r, M)$ then $(y_1, z_1(y_1)) \in \mathcal{S}_{\mathcal{Z}_1}(r, M)$ and $(y_2, z_2(y_2)) \in \mathcal{S}_{\mathcal{Z}_2}(r, M)$. Therefore, $z_1(y_1)$ is $L_{z_1 y}^{r, M}$ -Lipschitz continuous and $z_2(y_2)$ is $L_{z_2 y}^{r, M}$ -Lipschitz continuous by our assumption. Our result follows by defining $L_{zy}^{r, M} = L_{z_1 y}^{r, M} + L_{z_2 y}^{r, M}$.

Next we show that bi-convex g satisfies our assumption.

Lemma 24. Assumption 3 implies Assumption 9.

Proof. We first reclaim what we need to prove. For $(y, z) \in \mathcal{S}_{\mathcal{Z}}(r, M)$, $g(y, \bullet) + \iota_{\mathcal{Z}}(\bullet)$ has a unique global minimizer denoted as a function $z(y)$, such that $(y, z(y)) \in \mathcal{S}_{\mathcal{Z}}(\bar{r}_{zy}^{r, M}, M)$, in which is an $L_{zy}^{r, M}$ -Lipschitz continuous function with (r, M) -dependent parameters $\bar{r}_{zy}^{r, M}$ and $L_{zy}^{r, M}$.

The uniqueness of minimizer is due to the σ -strong convexity of g . Next, we know that $\mathcal{S}(r, M)$ is a compact set and $\|\nabla_z g(y, z)\|$ is a continuous function. Therefore, we have $\|\nabla_z g(y, z)\|$ is upper bounded on $\mathcal{S}(r, M)$ with the upper bound denoted as $L_{\nabla_z}^{r, M}$. We thus have the following

inequality:

$$\begin{aligned}
M &\stackrel{(1)}{\geq} g(y, z) \stackrel{(2)}{\geq} g(y, z(y)) \\
&\stackrel{(3)}{\geq} g(y, z) + \nabla_z g(y, z)^T (z(y) - z) + \frac{\sigma}{2} \|z(y) - z\|^2 \\
&\geq g(y, z) - L_{\nabla_z}^{r, M} \|z(y) - z\| + \frac{\sigma}{2} \|z(y) - z\|^2 \\
&\geq -L_{\nabla_z}^{r, M} \|z(y) - z\| + \frac{\sigma}{2} \|z(y) - z\|^2,
\end{aligned}$$

where (1) is due to our assumption, (2) is due to optimality, and (3) is due to σ -strong convexity. Solving the above quadratic inequality and we have:

$$\|z(y) - z\| \leq \left[\sqrt{2M\sigma + (L_{\nabla_z}^{r, M})^2} + L_{\nabla_z}^{r, M, N} \right] / \sigma \triangleq \delta_{\sigma}^{r, M}.$$

Therefore, we know that:

$$\begin{aligned}
\|(y, z) - (y, z(y))\| &\leq \|(y, z)\| + \|z - z(y)\| \\
&\leq r + \left[\sqrt{2M\sigma + (L_{\nabla_z}^{r, M})^2} + L_{\nabla_z}^{r, M, N} \right] / \sigma \triangleq \bar{r}_{zy}^{r, M}.
\end{aligned}$$

Now suppose $(y, z(y)) \in \mathcal{S}_{\mathcal{Z}}(r, M)$. By the inverse function theorem (Dontchev and Rockafellar 2009, Theorem 2F.7), we know that $0 \in N_{\mathcal{Z}}(z(y)) + \nabla_z g(y, z(y))$ where $z(y)$ has a single-valued localization in a neighborhood around $(y, z(y))$ with a Lipschitz modulus upper bounded by:

$$\begin{aligned}
&\| -\nabla_{zz} g(y, z(y))^{-1} \nabla_{zy} g(y, z(y)) \| \\
&\leq \| \nabla_{zz} g(y, z(y))^{-1} \| \| \nabla_{zy} g(y, z(y)) \| \stackrel{(1)}{\leq} L_g^{r, M} / \sigma \triangleq L_{zy}^{r, M},
\end{aligned}$$

where (1) is due to Lemma 10 and σ -strong convexity, which is our desired result.

Proof of Theorem 5. Combine Lemma 24 and Theorem 22.

We further show that the mass-spring energy $P_{k, l, \underline{l}}(y, z)$ satisfies our Assumption 9.

Lemma 25. $g = P_{k, l, \underline{l}}(y, z)$ and $\mathcal{Z} = \{z \mid \|z\| = 1\}$ satisfies Assumption 9.

Proof. Ohter parts being trivial, we only need to show Assumption 9 vi). First, we prove that $z(y) = (y_i - y_j) / \|y_i - y_j\|$ is the unique global minimizer. We have for an arbitrary unit z :

$$\begin{aligned}
&\|y_i - y_j - z\|^2 \\
&= \| [z(y)z(y)^T] (y_i - y_j - z) \|^2 + \\
&\quad \| [I - z(y)z(y)^T] (y_i - y_j - z) \|^2 \\
&= \|y_i - y_j - [z(y)z(y)^T] z\|^2 + \\
&\quad \| [I - z(y)z(y)^T] z \|^2 \\
&= \| \|y_i - y_j\| - \cos(\theta)l \|^2 + \|\sin(\theta)l\|^2 \\
&= \|y_i - y_j\|^2 + l^2 - 2l \cos(\theta) \|y_i - y_j\| \geq (\|y_i - y_j\| - l)^2,
\end{aligned}$$

where θ is the angle between $z(y)$ and z . From above formula, we can see that $z(y)$ is the unique optimal solution making $\cos(\theta) = 1$. Further, it can be easily verified that $z(y)$ is also the optimal solution to $-\log_{\epsilon}(n^T(y_i - y_j) - \underline{l})$.

Put together, $z(y)$ is the unique optimal solution minimizing $g = P_{k, l, \underline{l}}(y, z)$, proving the first part of Assumption 9 vi).

Suppose $(y, z) \in \mathcal{S}_{\mathcal{Z}}(r, M)$, then we have $\|y\| \leq r$ and $\|z(y)\| \leq 1$, so $\|(y, z(y))\| \leq r + 1 \triangleq \bar{r}_{zy}^{r, M}$, proving the second part of Assumption 9 vi).

From the solution $z(y)$ above, it is differentiable with respect to y with the derivative being:

$$\frac{\partial z(y)}{\partial y} = (I - z(y)z(y)^T, z(y)z(y)^T - I) \frac{1}{\|y_i - y_j\|}.$$

The above matrix has a norm upper bound of $\sqrt{2}/l \triangleq L_{zy}^{r, M}$, which is our desired Lipschitz modulus. We have thus proved the last part of Assumption 9 vi).

Finally, we show that the ARAP energy with strain limiting $P_F(y, z)$ satisfies our Assumption 9.

Lemma 26. If $P_{F, \epsilon}(y, z) \leq M$ then there exists some $\epsilon^M > 0$ and $\sigma^M > 0$ such that i) The sphere centered at y_0 with radius ϵ^M is contained in $\text{CH}(y_1, \dots, y_{d+1})$ and ii) $\sigma_k(F(y)) \geq \sigma^M$.

Proof. i) First note that all the terms in $P_{F, \epsilon}$ are positive, so each of them is smaller than M . Let us define $\text{CH}_i = \text{CH}(\{y_j \mid 0 < j \neq i\})$ as the face of the simplex element opposite to vertex y_i . We can claim that there exists some $\epsilon^M > 0$ such that $\text{dist}(y_0, \text{CH}_i) \geq \epsilon^M$. Because otherwise, the term $-\sum_{0 < j \neq i} \log_{\epsilon}(n_i^T(y_j - y_0)) = \infty$ models a separating plane between y_0 and CH_i , which is continuous and tends to infinity as $\text{dist}(y_0, \text{CH}_i) \rightarrow 0$ according to Liang et al. (2024), contradicting the fact that $P_{F, \epsilon}(y, z) \leq M$. We further have $y_0 \in \text{CH}(y_1, \dots, y_{d+1})$ because $0 = \iota_{\text{CH}(y_1, \dots, y_{d+1})}(y_0) \leq M$. This means the d -dimensional sphere centered at y_0 with radius ϵ^M is entirely contained the simplex element.

ii) We recall the following formula for calculating the deformation gradient $F(y)$:

$$\begin{aligned}
X^r &= (y_2^r - y_1^r, \dots, y_{d+1}^r - y_1^r) \\
X(y) &= (y_2 - y_1, \dots, y_{d+1} - y_1) \\
F(y) &= X(y)[X^r]^{-1},
\end{aligned}$$

where we use the superscript r to denote the rest pose. We know that $0 < \underline{\sigma}^r \leq \sigma_k(X^r)$ for some $0 < \underline{\sigma}^r$, i.e. X^r is non-singular by definition. By the barycentric coordinate interpolation formula, we know that for any $q \in \mathbb{R}^d$ with $\|q\| = \epsilon^M$, we have the barycentric coordinates $0 \leq \omega(q) \leq 1$, such that $y_0 + q = X(y)\omega(q)$. Now since $\iota_{\text{CH}_{y_1, \dots, d+1}}(y_0) = 0 \leq M$ by assumption, y_0 also has the barycentric coordinates, such that $y_0 = X(y)\omega(y_0)$. Subtracting the two equations and we have $\|q\| = \|X(y)[\omega(q) - \omega(y_0)]\|$, which immediately implies that $\sigma_k(X(y)) \geq \epsilon^M/d$. Combined with the fact that $\sigma_k(X^r) \geq \underline{\sigma}^r$, we conclude that $\sigma_k(F(y)) \geq \epsilon^M/(d\underline{\sigma}^r) = \underline{\sigma}^M$, which is our desired result.

Lemma 27. $g = P_{F, \epsilon}(y, z)$ and $\mathcal{Z} = \prod_{i=1}^{d+1} \{n_i \mid \|n_i\| \leq 1\} \times \text{SO}(3)$ satisfies Assumption 9.

Proof. We start by showing Assumption 9 vi). First note that when optimizing $z = (n_1, \dots, n_{d+1}, R)$, the n_i -subproblems and R -subproblem are separable. Suppose

$(y, z(y)) \in \mathcal{S}_{\mathcal{Z}}(r, M)$, then we have the analytic solution that $R = R(F)$ because R solves the orthogonal Procrustes problem Sorkine and Alexa (2007). The uniqueness of R follows from the uniqueness of polar decomposition. The uniqueness of n_i follows from the σ -strong convexity.

Now suppose $(y, z) \in \mathcal{S}_{\mathcal{Z}}(r, M)$, then we have $\|y\| \leq r$ and $\|z(y)\| \leq d + 1 + 9$, so $\|(y, z(y))\| \leq r + d + 10 \triangleq \bar{r}_{zy}^{r, M}$, proving the second part of Assumption 9 vi).

We still need to show that $z(y) = (n_i, R(F))$ is Lipschitz continuous when $(y, z(y)) \in (r, M)$. We only need to show that each n_i and $R(F)$ is Lipschitz continuous with respect to y and sum up the Lipschitz modulus to yield $L_{zy}^{r, M}$. The Lipschitz continuity of n_i follows by the same argument as in Lemma 24. From Lemma 26 ii), we know that $\sigma_k(F(y)) \geq \underline{\sigma}^M > 0$. It is known by Araki and Yamagami (1981) that $R(F)^T F$ is a Lipschitz continuous function of F with a modulus of $\sqrt{2}$. As a result, we have the following inequality:

$$\begin{aligned} & \|R(F(y))^T F(y) - R(F(y'))^T F(y')\|_F \\ & \leq \sqrt{2} \|F(y) - F(y')\|_F. \end{aligned}$$

We then have the following inequality:

$$\begin{aligned} & \|R(F(y))^T - R(F(y'))^T\|_F \\ & = \|(R(F(y))^T F(y) - R(F(y'))^T F(y))F(y)^{-1}\|_F \\ & = \|(R(F(y))^T F(y) - R(F(y'))^T F(y')) + \\ & \quad R(F(y'))^T F(y') - R(F(y'))^T F(y))F(y)^{-1}\|_F \\ & \leq \|(R(F(y))^T F(y) - R(F(y'))^T F(y'))\|_F \|F(y)^{-1}\|_F + \\ & \quad \|R(F(y'))^T\|_F \|F(y') - F(y)\|_F \|F(y)^{-1}\|_F \\ & \leq (\sqrt{2} + \sqrt{3}) \|F(y)^{-1}\|_F \|F(y') - F(y)\|_F \\ & \leq (\sqrt{2} + \sqrt{3}) \sqrt{\sum_{k=1}^d \sigma_k(F(y))^{-2} \|F(y') - F(y)\|_F} \\ & \leq (\sqrt{2} + \sqrt{3}) \sqrt{\sum_{k=1}^d (\underline{\sigma}^M)^{-2} \|F(y') - F(y)\|_F}, \end{aligned}$$

which shows that $R = R(F)$ is a Lipschitz-continuous function of F . We have thus proved the last part of Assumption 9 vi).

Finally, we turn to Assumption 9 i) and show that $P_{F, \epsilon}(y, z)$ is continuous in its entire domain, even though the indicator function terms $\iota_{\text{CH}(y_1, \dots, y_{d+1})}(y_0)$ and $\sum_{i=1}^3 \iota_{\sigma_k(F) > 0}(F(y))$ are not continuous in general. Let us suppose there is sequence $(y, z) \rightarrow (y^*, z^*)$. We first show that the term $\iota_{\text{CH}(y_1, \dots, y_{d+1})}(y_0)$ does not hinder continuity. In Case I, we assume $\iota_{\text{CH}(y_1, \dots, y_{d+1})}(y_0^*) = \infty$ then since $\text{CH}(y_1, \dots, y_{d+1})$ is a closed set, there is a neighborhood of (y^*, z^*) in which $\iota_{\text{CH}(y_1, \dots, y_{d+1})}(y_0) = \infty$. In Case II, we assume $\iota_{\text{CH}(y_1, \dots, y_{d+1})}(y_0^*) = 0$, then there are two sub-cases. In Case II.a, we assume $y_0^* \in \text{CH}^\circ(y_1, \dots, y_{d+1})$, then there is a neighborhood in which $\iota_{\text{CH}(y_1, \dots, y_{d+1})}(y_0) = 0$. In Case II.b, we assume $y_0^* \in \partial \text{CH}(y_1, \dots, y_{d+1})$, then by Lemma 26 i), we know that $P_{F, \epsilon}(y^*, z^*) = \infty$ because the plane separating term $-\sum_{0 < j \neq i} \log_\epsilon((n_i^*)^T (y_j^* - y_0^*)) = \infty$ for some i . And since this term is continuous, we have $\lim_{y, z} P_{F, \epsilon}(y, z) = \infty$.

Next, we show that the term $\sum_{i=1}^3 \iota_{\sigma_k(F) > 0}(F(y))$ does not hinder continuity. Again, we consider two cases. In Case I, we assume $\sum_{i=1}^3 \iota_{\sigma_k(F) > 0}(F(y^*)) = 0$, then since singular values are continuous functions of the matrix, we have a neighborhood in which $\sum_{i=1}^3 \iota_{\sigma_k(F) > 0}(F(y)) = 0$. In Case II, we assume $\sigma_k(F(y^*)) < 0$ for some k , then there is a neighborhood in which $\sum_{i=1}^3 \iota_{\sigma_k(F) > 0}(F(y)) = \infty$. In Case III, we have $\sigma_k(F(y^*)) = 0$ but $\sigma_k(F(y)) > 0$ for every (y, z) , then again we consider two sub-cases. In Case III.a, we assume $\iota_{\text{CH}(y_1, \dots, y_{d+1})}(y_0^*) = \infty$, then by the argument in the previous section, we have $\lim_{y, z} P_{F, \epsilon}(y, z) = \infty$. In Case III.b, we assume $\iota_{\text{CH}(y_1, \dots, y_{d+1})}(y_0^*) = 0$, then as argued in the previous section, we can again consider two sub-cases. In Case III.b1, $y_0^* \in \text{CH}^\circ(y_1, \dots, y_{d+1})$ and there is a neighborhood in which $\iota_{\text{CH}(y_1, \dots, y_{d+1})}(y_0) = 0$. But this is not possible by Lemma 26 ii). In Case III.b2, $y_0^* \in \partial \text{CH}(y_1, \dots, y_{d+1})$ then we again have $\lim_{y, z} P_{F, \epsilon}(y, z) = \infty$ as argued above. Thus all is proved.