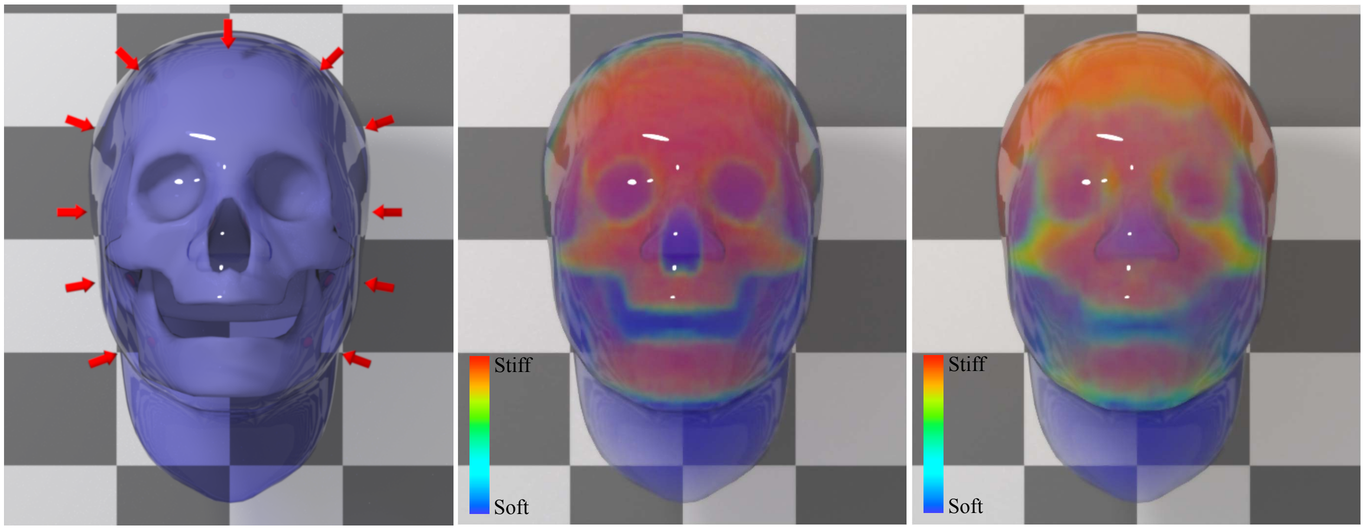
Inexact Descent Methods for Elastic Parameter Optimization

GUOWEI YAN, The Ohio State University, USA
WEI LI, University of Kentucky, USA
RUIGANG YANG, University of Kentucky, USA
HUAMIN WANG, The Ohio State University, USA



(a) The 3D object and the data samples (b) The ground truth (c) The result

Fig. 1. Given the synthetic data generated by multiple local contacts (such as those shown in red arrows) with the face, an inexact descent method estimates the shear moduli and detects the rigid skull underneath soft skin in 2.5 hours, as (c) shows. This example contains 150K tetrahedra, 150K elastic parameters, 168 data samples, and each sample uses 25K markers. Such a large example would take days or even weeks to handle by previous techniques.

Elastic parameter optimization has revealed its importance in 3D modeling, virtual reality, and additive manufacturing in recent years. Unfortunately, it is known to be computationally expensive, especially if there are many parameters and data samples. To address this challenge, we propose to introduce the inexactness into descent methods, by iteratively solving a forward simulation step and a parameter update step in an inexact manner. The development of such inexact descent methods is centered at two questions: 1) how accurate/inaccurate can the two steps be; and 2) what is the optimal way to implement an inexact descent method. The answers to these questions are in our convergence analysis, which proves the existence of relative error thresholds for the two inexact steps to ensure the convergence. This means we can simply solve each step by a fixed number of iterations, if the iterative solver is at least linearly convergent. While the use of the inexact idea speeds up many descent methods, we specifically favor a GPU-based one powered by state-of-the-art simulation techniques. Based on this method, we study a variety of implementation issues, including backtracking line search, initialization, regularization, and multiple data samples. We demonstrate the use of our inexact method in elasticity measurement and design applications. Our experiment shows the method is fast, reliable, memory-efficient, GPU-friendly, flexible with different elastic models, scalable to a large parameter space, and parallelizable for multiple data samples.

CCS Concepts: • **Computing methodologies → Physical simulation**;

Additional Key Words and Phrases: quasistatic simulation, constrained optimization, inexact method, heterogeneity, nonlinear elasticity

ACM Reference Format:
Guowei Yan, Wei Li, Ruigang Yang, and Huamin Wang. 2018. Inexact Descent Methods for Elastic Parameter Optimization. *ACM Trans. Graph.* 37, 6, Article 253 (November 2018), 14 pages. https://doi.org/10.1145/3272127.3275021

Authors' addresses: Guowei Yan, The Ohio State University, Columbus, OH, USA; Wei Li, University of Kentucky, Lexington, KY, USA; Ruigang Yang, University of Kentucky, Lexington, KY, USA; Huamin Wang, The Ohio State University, Columbus, OH, USA, whmin@cse.ohio-state.edu.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2018 Copyright held by the owner/author(s). Publication rights licensed to ACM.
0730-0301/2018/11-ART253 $15.00
https://doi.org/10.1145/3272127.3275021

1 INTRODUCTION

Accurate design, measurement, and simulation of real-world elastic materials is highly demanded by virtual reality applications. While elastic body simulation has been extensively studied in the past, elasticity measurement and design are much less explored. Given a set of measurement or user-specified data, elasticity measurement and design can be formulated into a parameter optimization problem, whose goal is to find the parameters that minimize the discrepancy between the simulation and the data. This optimization problem is computationally challenging, due to the highly nonlinear and heterogeneous force-displacement relationships of real-world elastic bodies. High nonlinearity increases the complexity of the optimization problem and deteriorates the convergence rate of an iterative solver. Meanwhile, high heterogeneity requires multiple material parameters and data samples, which further raises the computational burden. To address high nonlinearity and high heterogeneity, we need a fast, robust, and scalable optimization method that is suitable for a variety of elastic models.

Elastic parameter optimization is often solved by *descent methods* in computer graphics, mechanical engineering, biomechanics, and other areas, thanks to their flexibility and conceptual simplicity. In that case, a descent method typically needs to handle two steps in each of its iterations. In the forward step, it reaches quasistatic equilibrium using the current parameters; in the update step, it then modifies the parameters by classical methods, such as Nelder-Mead [Wang et al. 2015], Gauss-Newton [Bickel et al. 2009; Miguel et al. 2016], or L-BFGS [Casati et al. 2016]. Unfortunately, existing descent methods are notorious for their large computational costs, when they deal with highly nonlinear and heterogeneous bodies. Gradient-free methods and Gauss-Newton are too computationally expensive, as the parameter space gets large. Meanwhile, gradient descent and L-BFGS cannot converge fast and require too many forward steps.

In this paper, we explore the use of the inexact idea to accelerate descent methods. Our basic strategy is *to approximate the solutions to both steps in an inexpensive manner, rather than to solve them exactly.* If these approximations can reduce the step costs without significantly deteriorating the convergence rate, the resulting inexact method should be able to obtain a performance gain. While the inexact idea is simple, the success of developing inexact descent methods relies on two critical issues.

- *How accurate should the two steps be?* Making the steps too accurate would reduce the inexactness and the performance gain. On the other hand, making the steps too inaccurate would render the parameter update meaningless and potentially trigger the divergence issue.
- *What is the optimal way to implement the two steps?* While inexact methods are consistently faster than their exact counterparts, the optimal inexact method requires a good balance between the computational cost and the convergence rate, and it must take parallelization into consideration.

The inexact idea has been extensively studied in numerical literature before. However, an inexact descent method for elastic parameter optimization is one of its kind and it cannot be classified into any of the existing inexact methods, as far as we know. The main issue is that the forward step serves as the prerequisite of the update step, which fails if the forward step is inexact. This issue becomes more complicated, once we start to introduce the inexactness into the update step as well. Here we focus our research on both the theoretical and practical aspects of an inexact descent method. Our technical contributions can be summarized as follows.

- *Analysis.* We analyze the convergence of an inexact descent method. The analysis shows the method is guaranteed to be locally linear convergent, if the relative error drops below a fixed threshold in every inexact step. This conclusion is crucial to practical implementation, as it allows the inexactness to be controlled by a fixed number of iterations, if the iterative solver is at least linearly convergent.
- *Implementation.* Inexact descent methods can be implemented in different ways, according to their solvers. We argue that the optimal method should choose a fixed-point iterative solver with Chebyshev acceleration, which is highly compatible with the GPU [Wang and Yang 2016]. Compared with the optimal exact method using a direct solver with L-BFGS, our method runs orders-of-magnitude faster.
- *Improvements.* Finally, we study a series of practical topics to improve the convergence and the performance of our descent method. These topics include backtracking line search, initialization, regularization, and parallelization of multiple data samples.

In summary, we introduce the inexact idea into the development of descent methods for elastic parameter optimization, and we show that the inexactness can be safely achieved by fixed numbers of iterations. We then present a unique inexact method, which is fast, reliable, free of pre-computation or heavy memory usage, scalable to a large parameter space, GPU-friendly and parallelizable for multiple data samples. The method can flexibly handle a variety of hyperelastic models. The experiment reveals the use of our method in elasticity measurement (in Fig. 1) and interactive material design (in Fig. 12). Our concept is not restricted to elastic parameters and it is applicable to other inverse problems in the future.

2 OTHER RELATED WORK

Elastic body simulation. The pioneer work by Terzopoulos and colleagues [1987] has motivated graphics researchers to study elastic body simulation over the last thirty years. Early simulation techniques [O'Brien and Hodgins 1999; Teran et al. 2003] based on explicit time integration must choose small time steps to avoid the numerical instability issue. To take large time steps, simulators can use implicit integration and formulate a nonlinear system for every time step. Since it is too expensive to solve each nonlinear system exactly, Baraff and Witkin [1998] proposed to run a single Newton iteration only. This practice, commonly adopted by many simulators, has a risk of accumulating too much error over time, when the time step is large or the object deforms fast. A Newton iteration is essentially a linear system, whose matrix contains the Jacobian matrix of the elastic force. Teran and colleagues [2005] developed a matrix evaluation scheme for hyperelastic materials. Xu and collaborators [2015b] presented a matrix evaluation scheme for elastic models that can be formulated by principal stretches. We use their scheme in this work, thanks to its simplicity and generality.

The challenges in elastic body simulation are largely caused by the complexity of elastic models. An interesting idea is to replace elastic models partially or fully by positional constraints, also known as holonomic constraints [Müller 2008; Müller et al. 2005; Provot 1996; Thomaszewski et al. 2009; Wang et al. 2010]. The cost of doing this, however, is that the simulation loses its physical meaning and the stiffness becomes dependent of the iteration count. This problem can be addressed by formulating each positional constraint into an elastic energy, known as projective dynamics [Bouaziz et al. 2014; Liu et al. 2013]. Narain and colleagues [2016] demonstrated that projective dynamics can be considered as a special example of the alternating direction method of multipliers (ADMM). Wang [2015] and Fratarcangeli and collaborators [2016] explored the implementation of projective dynamics on the GPU, by using iterative solvers to solve each global iteration inexactly.

Elastic parameter optimization. In computer graphics, elastic parameter optimization is useful for elasticity measurement and design applications. For elasticity measurement, researchers were often more interested in data acquisition devices [Bhat et al. 2003; Bickel et al. 2010; Miguel et al. 2012; Pai et al. 2001; Wang et al. 2015, 2011], than optimization methods. Based on descent methods, their systems are typically restricted to sparsely sampled elastic parameters [Bickel et al. 2009; Miguel et al. 2016; Wang et al. 2015], and/or linear co-rotational models. For interactive elasticity design, the computational cost can become a serious issue. Bickel and collaborators [2010] proposed to model elastic material distribution in layers. Xu and colleagues [2015a] suggested to design material distribution in a reduced subspace. Recently, Miguel and collaborators [2016] advocated the use of energy constraints for modeling more plausible elastic behaviors, but they did not specifically address the computational burden.

In civil and mechanical engineering, researchers have studied elasticity parameter optimization for identifying cracks and defects within solid objects. Since these objects are highly stiff, their research [Bonnet and Constantinescu 2005; Khodadad and Ardakani 2009] was focused on small deformations and linear elastic models. Our work is more closely related to elastography in biomechanics, which tries to identify tumors within soft tissues through elastic parameter optimization. Lonbani [2010] and Doyley [2012] surveyed optimization techniques in this field, including descent methods. Berger [2009] studied elastic parameter optimization with boundary element discretization. Gockenbach and colleagues [2015] studied the use of proximal methods to solve a residual force formulation, under the assumption that the displacement field is densely sampled. O'Hagan, Samani and collaborators [2011; 2008] proposed to solve the update step of a descent method by a slope variation approach or the Nelder-Mead method. Their technique avoids expensive gradient and matrix evaluations, at the cost of extra difficulty in handling heterogeneous solids.

In summary, researchers have applied descent methods in a wide range of elastic parameter optimization applications. However, they were forced to sacrifice nonlinearity or heterogeneity in these methods, due to the computational burden. By adopting the inexact idea, our research allows descent methods to reach their full potentials.

Elastic shape design. Our research is also closely related to elastic shape design, which treats the reference shape as the optimization variables. This field has been increasingly active in computer graphics, due to its potential use in additive fabrication of elastic objects. Over the past few years, researchers have studied a variety of topics, including flexible rods [Pérez et al. 2015], an asymptotic method [Chen et al. 2014], garments and sewing patterns [Casati et al. 2016; Umetani et al. 2011; Wang 2018], design objectives and tools [Megaro et al. 2017], Kirchhoff-Plateau surfaces [Pérez et al. 2017], and silicone composites [Zehnder et al. 2017]. While many of these works can potentially benefit from the use of inexact descent methods, we do not investigate them in this work, because the inverse problem alone is only one of the problems involved in elastic shape design.

Inexact methods. The inexact idea is not new and it has been widely explored by the numerical optimization community. For most of the numerical methods, researchers have developed their inexact counterparts, including inexact Newton methods [Dembo et al. 1982; Eisenstat and Walker 1994], inexact Quasi-Newton methods [Bergamaschi et al. 2001; Birgin et al. 2003], inexact Newton-Dogleg methods [Pawlowski et al. 2008], inexact SQP methods [Byrd et al. 2008], and inexact proximal point methods [Burachik and Dutta 2010; Solodov and Svaiter 2001]. It should be noted that the descent methods discussed in this work look similar to many splitting methods, such as the alternating direction method of multipliers (ADMM) and proximal point methods, but they are fundamentally different due to the strong dependency of descent methods on the exactness of the forward simulation step. Because of that, inexact descent methods are fundamentally different from existing inexact splitting methods as well.

Despite their popularity in numerical optimization, inexact methods are relatively uncommon in graphics research. Wang [2015] described the inexact idea when he developed a GPU-based accelerator for projective dynamics and position-based dynamics. Fratarcangeli and colleagues [2016] strengthened this technique by multi-color Gauss-Seidel solvers. Wang and Yang [2016] applied the same idea to the simulation of hyperelastic bodies, which can be considered as a special example of inexact Newton methods.

The use of inexact methods in solving inverse problems, especially elastic parameter optimization, is quite limited. Lund and colleagues [2003] used an inexact Jacobian matrix in sensitivity analysis for solving an inverse fluid shape design problem. Haber and colleagues [2004] proposed to solve forward simulation inexactly to speed up the inversion of electromagnetic data. Unlike descent methods, their method is based on sequential quadratic programming using the primal-dual formulation of constrained optimization. Quirynen and collaborators [2017] considered inexact forward simulation with the primal-dual formulation as well, and they developed an inexact Newton-type method with iterative sensitivities for preserving local convergence properties. Recently, Wang [2018] proposed a unique inexact descent method for solving the inverse sewing pattern design problem, by applying inexact forward simulation with the primal formulation. While his method has demonstrated its high performance, it comes with no convergence analysis or justification. In this work, we not only introduce the inexactness into both forward simulation and sensitivity analysis, but also systematically explore the convergence issue caused by the inexactness. Our research enables descent methods to achieve much higher performance.

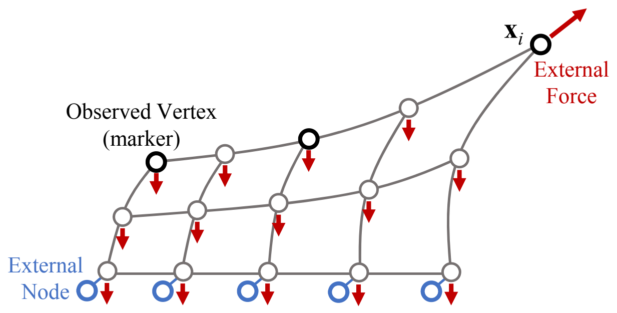


Fig. 2. A data sample. This sample contains three observed vertices (in black) and all of the external forces (in red) applied on the vertices, most of which are gravity forces. The spring forces attaching bottom vertices to external nodes (in blue) are treated as unknown internal forces.

3 PROBLEM DEFINITION

The goal of elastic parameter optimization is to find the optimal elastic parameters, using which the simulation becomes consistent with the data. For simplicity, we will focus our discussion mostly on a single data sample in this section. The extension to multiple samples will be studied in Subsection 5.5.

Given an elastic body whose elastic parameters are unknown, we define a data sample as a set of observed vertices (i.e., markers) and all of the external forces applied on the vertices, as Fig. 2 shows. We assume that certain vertices are attached to external boundary nodes by stiff springs and we treat these spring forces as unknown internal forces. This is slightly more straightforward to implement than removing the degrees of freedoms caused by fixed vertices. We assume that data samples are collected in quasistatic equilibrium.

Let $\mathbf{x} \in \mathbb{R}^{3N}$ be the simulated vertex vector satisfying the quasistatic condition: $\mathbf{f}(\mathbf{k}, \mathbf{x}) = \mathbf{0}$, in which $\mathbf{k} \in \mathbb{R}^M$ is a set of M elastic parameters and $\mathbf{f}$ is the conservative force vector as a function of $\mathbf{k}$ and $\mathbf{x}$. Let $\mathbf{y} \in \mathbb{R}^n$ be the vector containing observed markers in the data sample. The goal is to find $\mathbf{k}$, such that: $\mathbf{Sx} = \mathbf{y}$ and $\mathbf{f}(\mathbf{k}, \mathbf{x}) = \mathbf{0}$, where $\mathsf{S} \in \mathbb{R}^{n\times 3N}$ selects observed coordinates. There are three typical ways to formulate this problem mathematically.

Nonlinear least squares. Perhaps the most straightforward way is to formulate the problem in a nonlinear least squares (NLS) fashion:

$$\{\mathbf{x}, \mathbf{k}\} = \arg\min_{\{\mathbf{x},\mathbf{k}\}} \frac{1}{2}\left\{\|\mathsf{S}\mathbf{x} - \mathbf{y}\|^2 + \|\mathbf{f}(\mathbf{k}, \mathbf{x})\|^2\right\} + R(\mathbf{k}), \tag{1}$$

in which $R(\mathbf{k})$ is a regularization term protecting the problem from being under-constrained. Unfortunately, Equation 1 is ill-conditioned and contains too many local minima, which makes typical NLS methods, such as Gauss-Newton, difficult to find meaningful solutions.

Residual error minimization. Alternatively, parameter optimization can be formulated as the minimization of the quasistatic residual [Becker and Teschner 2007]:

$$\mathbf{k} = \arg\min_{\mathbf{k}} \frac{1}{2}\left\{\|\mathsf{S}\mathbf{f}(\mathbf{k}, \mathbf{x})\|^2\right\} + R(\mathbf{k}), \tag{2}$$

in which $\mathbf{x}$ is the quasistatic equilibrium state subject to boundary constraints: $\mathbf{Sx} = \mathbf{y}$. The problem with this formulation is that it is also sensitive to the local minimum issue. In particular, a descent method applied to this formulation prefers to decrease the stiffness locally near the observations first. This quickly reduces the objective, but the final result can be far away from a global solution.

Output least squares. Eventually we choose to formulate elastic parameter optimization by minimizing the difference between the observation and the simulation output:

$$\mathbf{k} = \arg\min_{\mathbf{k}} \frac{1}{2}\left\{\|\mathsf{S}\mathbf{x} - \mathbf{y}\|^2\right\} + R(\mathbf{k}), \quad \text{subject to} \;\; \mathbf{f}(\mathbf{k}, \mathbf{x}) = \mathbf{0}. \tag{3}$$

A classical way of solving constrained optimization is to introduce Lagrangian multipliers and solve the resulting KKT conditions as a nonlinear system. The main issue is that the Jacobian of the KKT conditions, i.e., the Hessian of the Lagrangian function, is indefinite. This causes many nonlinear solvers to be ineffective or inefficient. Finding a suitable merit function without undermining the convergence rate [Nocedal and Wright 2006] is another difficult problem. For example, using Newton's method to solve the KKT conditions, known as sequential quadratic programming (SQP), performs much worse than descent methods as Fig. 5d shows.

Existence, uniqueness, and local minima. Before we begin our study on descent methods, we would like to discuss the limitations of the output least squares problem in Equation 3. Being in a constrained nonlinear least squares form, this problem is guaranteed to have a global minimum. However, the global minimum may not be meaningful, i.e., not satisfying $\mathbf{Sx} \approx \mathbf{y}$. The global minimum may not be unique either, depending on how the regularization term is defined. Perhaps the biggest limitation of Equation 3 is the local minima. While it contains substantially fewer local minima than other formulations, it is still not free of them. One of the reasons is because the elastic energy may not be convex, e.g., when the body bends under compression.

We would like to emphasize that these issues are intrinsic to elastic parameter optimization and they exist in almost every previous work. In this research, we do not focus our study on solving these issues. Instead, we assume that elastic parameters have been properly initialized, and our goal is to develop a fast method for finding the corresponding local minimum. In our experimental settings, data samples are collected from local contacts and the local minimum issue is not severe in practice.

4 METHODOLOGY

Our research is focused on finding an efficient and robust way to solve the problem formulated in Equation 3. To begin with, we will describe descent methods and explain why they are inefficient. Next in Subsection 4.2, we will present the idea of inexact descent methods and provide the error conditions for their convergence. These conditions are critical to the performances of their practical implementations discussed later in Section 5.

4.1 Descent methods

For elastic parameter optimization, the basic idea behind a descent method is to eliminate the constraint and treat $\mathbf{x}$ as a function of $\mathbf{k}$: $\mathbf{x} = \mathcal{X}(\mathbf{k})$. The resulting objective becomes unconstrained:

$$C(\mathbf{k}) = \frac{1}{2}\left\{\|\mathsf{S} \cdot \mathcal{X}(\mathbf{k}) - \mathbf{y}\|^2 + R(\mathbf{k})\right\}. \tag{4}$$

ALGORITHM 1: Elastic Parameter Optimization

```
Input: The selection matrix S, the observation y, the initial parameters
       $\mathbf{k}^{(0)}$, the initial step length $S$, and the numbers of inner
       iterations $A$ and $B$.
Output: The final parameters $\mathbf{k}$.
$\mathbf{x}^{(0)} \leftarrow \mathcal{X}(\mathbf{k}^{(0)})$; $s^{(-1)} \leftarrow S$; $l \leftarrow 0$;
repeat
    $\mathbf{J_k} \leftarrow \partial \mathbf{f}(\mathbf{k}^{(l)}, \mathbf{x}^{(l)})/\partial \mathbf{k}$;
    $\mathbf{J_x} \leftarrow \partial \mathbf{f}(\mathbf{k}^{(l)}, \mathbf{x}^{(l)})/\partial \mathbf{x}$;
    $\mathbf{a}^{(l)} \leftarrow$ Linear_Solver($\mathbf{J}_\mathbf{x}^\mathsf{T}, \mathbf{S}^\mathsf{T}(\mathbf{S}\mathbf{x}^{(l)} - \mathbf{y}), B$);    // inner loop
    $\mathbf{g}^{(l)} \leftarrow -\mathbf{J}_\mathbf{k}^\mathsf{T}\mathbf{a}^{(l)} + \nabla R(\mathbf{k}^{(l)})$;
    $s^{(l)} \leftarrow \min(S, s^{(l-1)}/\beta)$;
    repeat
        $\mathbf{k}^{(l+1)} \leftarrow \mathbf{k}^{(l)} - s^{(l)}\mathbf{g}^{(l)}$;
        $\mathbf{x}^{(l+1)} \leftarrow$ Simulator($\mathbf{x}^{(l)}, \mathbf{k}^{(l+1)}, A$);    // inner loop
        $s^{(l)} \leftarrow \beta s^{(l)}$;
    until Some sufficient decrease condition is satisfied;
    $l \leftarrow l + 1$;
until $\|\mathbf{f}(\mathbf{k}^{(l)}, \mathbf{x}^{(l)})\| < \epsilon_0$ and $\|\mathbf{g}^{(l)}\| < \epsilon_1$;
return $\mathbf{k}^{(l+1)}$;
```

Its gradient, used as the source of the search direction, is:

$$\nabla C(\mathbf{k}) = \left(\frac{\partial C(\mathbf{k})}{\partial \mathbf{k}}\right)^\mathsf{T} = \left(\frac{\partial \mathcal{X}(\mathbf{k})}{\partial \mathbf{k}}\right)^\mathsf{T} \mathbf{S}^\mathsf{T}(\mathbf{S}\cdot\mathcal{X}(\mathbf{k}) - \mathbf{y}) + \nabla R(\mathbf{k}). \quad (5)$$

Since $\mathbf{f}(\mathbf{k}, \mathcal{X}(\mathbf{k})) \equiv \mathbf{0}$, we apply the implicit function theorem to establish the sensitivity between $\mathcal{X}(\mathbf{k})$ and $\mathbf{k}$:

$$\frac{\partial \mathcal{X}(\mathbf{k})}{\partial \mathbf{k}} = -\mathbf{J}_\mathbf{x}^{-1}(\mathbf{k}, \mathcal{X}(\mathbf{k})) \cdot \mathbf{J_k}(\mathbf{k}, \mathcal{X}(\mathbf{k})), \quad (6)$$

in which $\mathbf{J_x} = \partial \mathbf{f}/\partial \mathbf{x} \in \mathbb{R}^{3N\times 3N}$ and $\mathbf{J_k} = \partial \mathbf{f}/\partial \mathbf{k} \in \mathbb{R}^{3N\times M}$ are the Jacobian matrices of $\mathbf{f}$. By formulating two functions:

$$\begin{aligned} \mathcal{G}(\mathbf{k}, \mathbf{x}, \mathbf{a}) &= -\mathbf{J}_\mathbf{k}^\mathsf{T}(\mathbf{k}, \mathbf{x}) \cdot \mathbf{a} + \nabla R(\mathbf{k}), \\ \mathcal{A}(\mathbf{k}, \mathbf{x}) &= \mathbf{J}_\mathbf{x}^{-\mathsf{T}}(\mathbf{k}, \mathbf{x}) \cdot \mathbf{S}^\mathsf{T}(\mathbf{S}\mathbf{x} - \mathbf{y}), \end{aligned} \quad (7)$$

we define the gradient as $\nabla C(\mathbf{k}) = \mathcal{G}(\mathbf{k}, \mathcal{X}(\mathbf{k}), \mathcal{A}(\mathbf{k}, \mathcal{X}(\mathbf{k})))$, in which $\mathcal{A}(\mathbf{k}, \mathcal{X}(\mathbf{k}))$ is known as the *adjoint state*.

According to the above description, we process every iteration of a descent method in two steps. In the forward step, it calculates the quasistatic equilibrium state $\mathcal{X}(\mathbf{k})$ using the current $\mathbf{k}$ first. In the update step, it then computes $\nabla C(\mathbf{k})$ and uses that to obtain the search direction for updating $\mathbf{k}$. In this paper, we simply treat the opposite of the gradient as the search direction, known as *gradient descent*. While the inexact idea is not restricted to gradient descent, other methods, such as Gauss-Newton [Bickel et al. 2009; Miguel et al. 2016] and Newton-type methods, often require intensive matrix evaluation of $\mathbf{J}_\mathbf{k}^\mathsf{T}\mathbf{J}_\mathbf{x}^{-\mathsf{T}}\mathbf{J}_\mathbf{x}^{-1}\mathbf{J_k}$, which is unaffordable if the parameter space is large. For instance, MKL PARDISO needs 24.3ms to solve a linear system for the teddy bear example in Fig. 4. The example contains 49K parameters, and the calculation of the whole matrix would take 20 minutes per iteration per data sample. In contrast, gradient descent is free of intensive matrix evaluation.

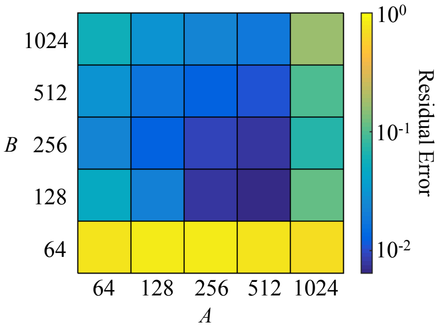


Fig. 3. The performance of an inexact method when using different A and B values. This plot shows that reducing A and B improves the performance, before the method hits the divergence issue. Here the errors are reported from the teddy bear example, after running the same method for 10 minutes.

4.2 Inexact Descent Methods

A descent method contains two computational bottlenecks: quasistatic simulation involved in the forward step and the linear system for calculating the adjoint state in the update step, both of which need to run multiple inner iterations. To incorporate the inexactness into such a method, the natural idea is to simply restrict the numbers of inner iterations A and B, as shown in Algorithm 1. By reducing the cost per outer iteration without significantly undermining the convergence rate, an inexact descent method is expected to achieve higher performance than its exact counterpart. Fig. 3 further shows that an inexact method runs faster as A and B become smaller, until the divergence issue occurs. Here two key questions are:

- How to adjust A and B to their minimal values without causing divergence?
- How often do A and B need to be adjusted?

The second question is of particular importance, because if the adjustment is frequent, its overhead could potentially destroy the performance gain brought by the inexactness.

Fortunately, we will show in the following analysis that there exists lower bounds on A and B, for an inexact descent method to converge safely as a standard line search method. This allows us to monotonically and dynamically increase A and B using a backtracking strategy during optimization. In our implementation, we carry out this adjustment together with step length adjustment, using long-range backtracking as discussed later in Subsection 5.2. The total adjustment overhead is bounded, since A and B are bounded.

Table 1 summarizes the notations to be used in the analysis.

4.2.1 Error conditions. To begin with, we would like to formulate the conditions for an inexact descent method to converge. Unlike other methods, especially proximal point methods [Burachik and Dutta 2010; Solodov and Svaiter 2001], a descent method relies on the implicit function theorem to bridge between its two steps. Since the theorem becomes invalid if the forward step is inexact, we cannot analyze the inexactness of the two steps in an independent fashion. Instead, we establish our analysis by viewing a descent method fundamentally as a line search method. For a line search method to work, we have to make sure that the calculated inexact gradient $\mathbf{g}^{(l)} = \mathcal{G}(\mathbf{k}^{(l)}, \mathbf{x}^{(l)}, \mathbf{a}^{(l)})$, which uses an inexact quasistatic

Table 1. Notations of exact and inexact variables. Our convergence analysis uses the following notations in Subsection 4.2 and appendices.

Description	Exact Variable	Inexact Variable
Parameters	$\mathbf{k}$	-
Quasistatic state	$\mathcal{X}(\mathbf{k})$	$\mathbf{x}$
Adjoint state	$\mathcal{A}(\mathbf{k}, \mathcal{X}(\mathbf{k}))$	$\mathcal{A}(\mathbf{k}, \mathbf{x})$ $\mathbf{a}$
Gradient	$\nabla C(\mathbf{k}) =$ $\mathcal{G}(\mathbf{k}, \mathcal{X}(\mathbf{k}), \mathcal{A}(\mathbf{k}, \mathcal{X}(\mathbf{k})))$	$\mathcal{G}(\mathbf{k}, \mathbf{x}, \mathbf{a})$ $\mathcal{G}(\mathbf{k}, \mathbf{x}, \mathcal{A}(\mathbf{k}, \mathbf{x}))$

state $\mathbf{x}^{(l)}$ and an inexact adjoint state $\mathbf{a}^{(l)}$, is always a valid descent direction:

$$-\mathbf{g}^{(l)} \cdot \nabla C(\mathbf{k}^{(l)}) < 0. \tag{8}$$

Suppose that $\mathcal{G}(\mathbf{k}, \mathbf{x}, \mathcal{A}(\mathbf{k}, \mathbf{x}))$ is Lipschitz continuous with respect to $\mathbf{x}$, and $\mathcal{G}(\mathbf{k}, \mathbf{x}, \mathbf{a})$ is Lipschitz continuous with respect to $\mathbf{a}$. Equation 8 is satisfied, if the errors of the two steps meet the following conditions (Lemma A.1, Appendix A):

$$\begin{cases} \left\| \mathbf{x}^{(l)} - \mathcal{X}(\mathbf{k}^{(l)}) \right\| < \frac{1}{2L_{\text{gx}}} \left\| \nabla C(\mathbf{k}^{(l)}) \right\|, \\ \left\| \mathbf{a}^{(l)} - \mathcal{A}(\mathbf{k}^{(l)}, \mathbf{x}^{(l)}) \right\| < \frac{1}{2L_{\text{ga}}} \left\| \nabla C(\mathbf{k}^{(l)}) \right\|, \end{cases} \tag{9}$$

where L_{gx} and L_{ga} are the Lipschitz constants. Intuitively, if the errors are small, the inexact gradient must be close to the exact gradient and it must still be a valid descent direction.

4.2.2 Relative error conditions. An inexact method cannot enforce the error conditions in Equation 9 directly, since it cannot calculate $\nabla C(\mathbf{k}^{(l)})$, $\mathcal{X}(\mathbf{k}^{(l)})$, or $\mathcal{A}(\mathbf{k}^{(l)}, \mathbf{x}^{(l)})$ exactly. Instead, we show in the following theorems that these conditions can be maintained through the optimization process, by enforcing thresholds on the decreases of relative errors between two consecutive iterations. We name these thresholds as relative error conditions.

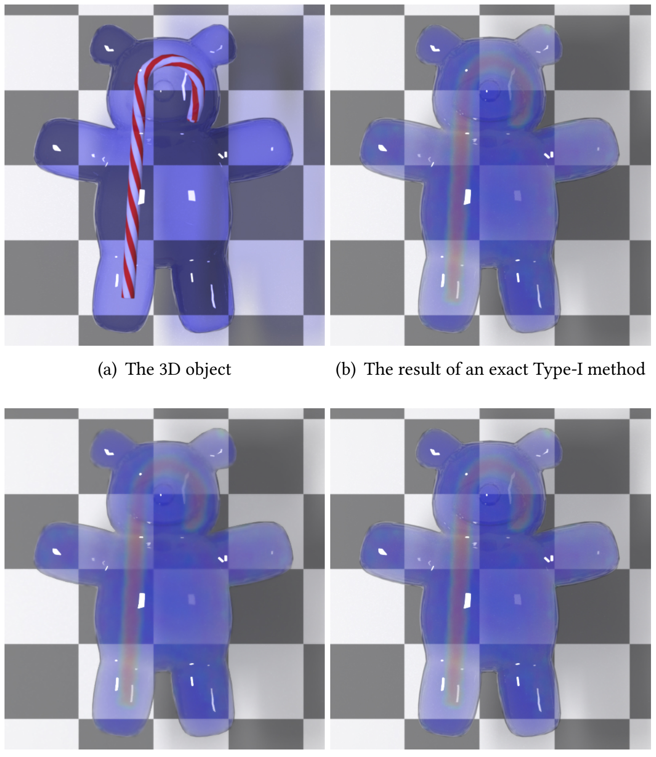
THEOREM 4.1. *Let $\mathcal{G}(\mathbf{k}, \mathbf{x}, \mathcal{A}(\mathbf{k}, \mathbf{x}))$ be Lipschitz continuous with respect to $\mathbf{x}$, $\mathcal{G}(\mathbf{k}, \mathbf{x}, \mathbf{a})$ be Lipschitz continuous with respect to $\mathbf{a}$, and $\nabla C(\mathbf{k})$ and $\mathcal{X}(\mathbf{k})$ be Lipschitz continuous with respect to $\mathbf{k}$. If the error conditions are met in the l-th outer iteration, there exists a constant μ, such that $\left\| \mathbf{x}^{(l+1)} - \mathcal{X}(\mathbf{k}^{(l+1)}) \right\| \leq \frac{1}{2L_{\text{gx}}} \left\| \nabla C(\mathbf{k}^{(l+1)}) \right\|$, if $\mathbf{x}^{(l+1)}$ satisfies $\left\| \mathbf{x}^{(l+1)} - \mathcal{X}(\mathbf{k}^{(l+1)}) \right\| \leq \mu \left\| \mathbf{x}^{(l)} - \mathcal{X}(\mathbf{k}^{(l+1)}) \right\|$.*

PROOF. See Theorem A.2, Appendix A. □

THEOREM 4.2. *Let $\mathcal{G}(\mathbf{k}, \mathbf{x}, \mathcal{A}(\mathbf{k}, \mathbf{x}))$ be Lipschitz continuous with respect to $\mathbf{x}$, $\mathcal{G}(\mathbf{k}, \mathbf{x}, \mathbf{a})$ be Lipschitz continuous with respect to $\mathbf{a}$, $\nabla C(\mathbf{k})$, $\mathcal{X}(\mathbf{k})$ and $\mathcal{A}(\mathbf{k}, \mathbf{x})$ be Lipschitz continuous as well. If the error conditions are met in the l-th outer iteration and the first error condition is met in the next outer iteration, there exists a constant ξ, such that $\left\| \mathbf{a}^{(l+1)} - \mathcal{A}(\mathbf{k}^{(l+1)}, \mathbf{x}^{(l+1)}) \right\| \leq \frac{1}{2L_{\text{ga}}} \left\| \nabla C(\mathbf{k}^{(l+1)}) \right\|$, if $\mathbf{a}^{(l+1)}$ satisfies $\left\| \mathbf{a}^{(l+1)} - \mathcal{A}(\mathbf{k}^{(l+1)}, \mathbf{x}^{(l+1)}) \right\| \leq \xi \left\| \mathbf{a}^{(l)} - \mathcal{A}(\mathbf{k}^{(l+1)}, \mathbf{x}^{(l+1)}) \right\|$.*

PROOF. See Theorem A.3, Appendix A. □

Since most solvers are at least linearly convergent, Theorem 4.1 and 4.2 are equivalent to saying that lower bounds on the numbers of inner iterations exist, for keeping the calculated gradient as a valid descent direction. Unfortunately, while these lower bounds are derivable from the proofs, they are not easily calculable in practice. This is why we apply a backtracking strategy to adjust the numbers of inner iterations, A and B, in a monotonic and dynamic fashion, as discussed later in Subsection 5.2.



(a) The 3D object (b) The result of an exact Type-I method

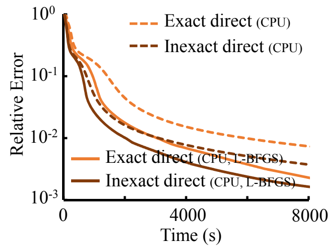
(c) The result of an inexact Type-II method (d) The result of an inexact Type-III method

Fig. 4. The teddy bear example. In this example, we generate synthetic data samples by touching the surface of a teddy bear at 20 contact points. This figure shows that all of the methods produce nearly the same heterogeneous result, which reveals the rigid candy hidden inside of the bear. *In the rest of the paper, we use this example for evaluation by default.*

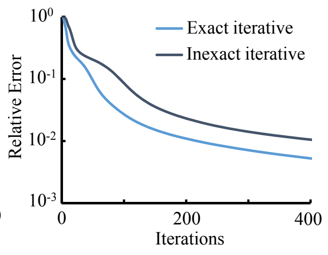
Like other line search methods, inexact descent methods are globally linear convergent, if the line search is exact ([Nocedal and Wright 2006], Theorem 3.4). In reality, exact line search is too expensive and we use backtracking line search instead. Appendix B proves that the resulting inexact methods preserve the locally linear convergence property.

5 PRACTICAL IMPLEMENTATIONS

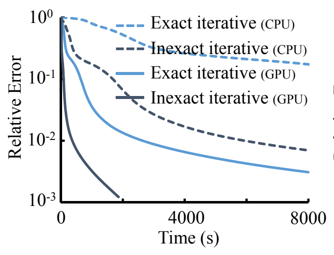
In this section, we would like to investigate practical issues related to the implementations of inexact descent methods. First we present three types of basic descent methods, whose steps use direct and iterative solvers respectively. Among them, we advocate a GPU-based implementation with fixed-point iterative solvers and Chebyshev acceleration. Based on this unique method, we then explore a number of additional issues, such as backtracking line search, initialization, regularization, and multiple data samples.



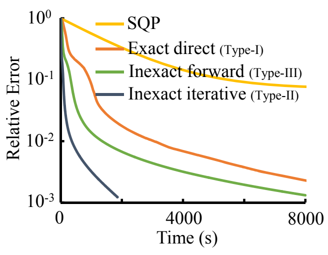
(a) The performance of Type-I methods



(b) The convergence of Type-II methods



(c) The performance of Type-II methods



(d) The performance of descent methods

Fig. 5. The performance of descent methods in the teddy bear example. These methods differ in how the two steps are solved. Compared with other methods, our inexact Type-II method runs the fastest thanks to a thorough inexact implementation on the GPU.

5.1 Basic Methods

Descent methods can be implemented in different ways, according to the solvers used by the two steps. Here we consider three types of descent methods. As shown in Fig. 4 and 5, their difference is mostly in performance, not optimization outcomes.

Type-I methods use direct LU factorization to solve the linear systems involved in both steps. In that case, the update step is exact, while the forward step can be made inexact by using fewer Newton iterations, each of which leads to a linear system. Fig. 5a shows that inexact Type-I methods are consistently faster than exact ones, with and without L-BFGS acceleration. Here we implement Type-I methods on the CPU only, since the performance of direct LU factorization on the CPU is about the same as that on the GPU.

Type-II methods use iterative solvers to solve the linear system in the update step and the nonlinear system in the forward step. Specifically, we implement the methods using the Jacobi iterative solver with Chebyshev acceleration [Golub and Van Loan 2012], in which case the forward step is equivalent to the simulator proposed by Wang and Yang [2016]. This implementation has a significantly lower cost per iteration on the GPU, thanks to high parallelizability of the Jacobi solver and Chebyshev acceleration. Fig. 5c shows that the resulting inexact Type-II method reaches a small error fast, even though it requires more iterations than the exact Type-II method as shown in Fig. 5b.

Finally, Type-III methods use direct LU factorization in the exact update step and a nonlinear iterative solver in the inexact forward step. As before, we choose the Jacobi solver with Chebyshev acceleration as our nonlinear iterative solver. Fig. 5d shows that an inexact Type-III method, similar to the method developed by Wang [2018], outperforms the exact Type-I method. But it is still slower than the inexact Type-II method, due to less inexactness.

We would like to emphasize that the aforementioned methods are only a subset of the descent methods that can benefit from adopting the inexact idea. For example, Type-II methods can use other fixed-point solvers, such as multi-color Gauss-Seidel [Fratarcangeli et al. 2016]. The methods can also use other solvers for the two steps. Instead of enumerating all of the possibilities, we discuss the three specific types, because they are sufficient for demonstrating the importance of the inexactness. First, the inexactness improves the performance of descent methods, regardless of their solvers. Second, the inexactness enables the effective use of iterative solvers on the GPU. In contrast, exact methods perform slowly with iterative solvers, and they should choose direct solvers instead. In the rest of this paper, we name the GPU-based Type-II method with the Jacobi solver and Chebyshev acceleration as our method by default.

5.2 Backtracking Line Search

The analysis in Section 4.2 assumes that we can obtain the step length by evaluating the Armijo condition exactly:

$$C(\mathbf{k}^{(l)} - s^{(l)}\mathbf{g}^{(l)}) < C(\mathbf{k}^{(l)}) - \alpha s^{(l)}\mathbf{g}^{(l)} \cdot \nabla C(\mathbf{k}^{(l)}), \tag{10}$$

which specifies a sufficient decrease of the objective function from one iteration to another. However, the condition cannot be evaluated exactly without using exact solvers, which would violate the inexact philosophy. A natural idea is to use an inexact Armijo condition:

$$\tilde{C}(\mathbf{k}^{(l+1)}, \mathbf{x}^{(l+1)}) < \tilde{C}(\mathbf{k}^{(l)}, \mathbf{x}^{(l)}) - \alpha s^{(l)} \left\|\mathbf{g}^{(l)}\right\|^2, \tag{11}$$

for $\tilde{C}(\mathbf{k}, \mathbf{x}) = \frac{1}{2}\left\{\|\mathbf{Sx} - \mathbf{y}\|^2 + R(\mathbf{k})\right\}$. Since $\tilde{C}(\mathbf{k}, \mathbf{x})$ is different from $C(\mathbf{k})$, the method potentially suffers from slow convergence or divergence. Slow convergence occurs, when the exact condition is satisfied but the inexact condition is not. As a result, the step length becomes unnecessarily small due to the inexact condition. In constrained optimization, a similar issue is known as the Maratos effect [Maratos 1978], in which enforcing strict decrease of a merit function inhibits the convergence rate. On the other hand, divergence happens, when the inexact condition is satisfied but the exact condition is not. This issue can be demonstrated as oscillation around the solution, as the step length causes overshooting.

Our solution is a loosely defined inexact Armijo condition:

$$\tilde{C}(\mathbf{k}^{(l+1)}, \mathbf{x}^{(l+1)}) < \tilde{C}(\mathbf{k}^{(l)}, \mathbf{x}^{(l)}) - \alpha s^{(l)} \left\|\mathbf{g}^{(l)}\right\|^2 + \gamma \left\|\mathbf{g}^{(l)}\right\| \left\|\mathbf{Sx}^{(l)} - \mathbf{y}\right\|, \tag{12}$$

where γ is a looseness constant. In every iteration, we initialize the step length by the one used in the previous iteration, and keep adjusting it until it satisfies Equation 12. The rationale behind Equation 12 comes from an estimate of the error:

$$\begin{aligned}&\frac{1}{2}\left(\|\mathbf{Sx} - \mathbf{y}\|^2 - \|\mathbf{S}\mathcal{X}(\mathbf{k}) - \mathbf{y}\|^2\right)\\ &\quad\approx (\mathbf{Sx} - \mathbf{S}\mathcal{X}(\mathbf{k})) \cdot (\mathbf{Sx} - \mathbf{y}) \leq \frac{\|\mathbf{S}\|}{2L_{\mathbf{gx}}} \|\nabla C(\mathbf{k})\| \, \|\mathbf{Sx} - \mathbf{y}\|.\end{aligned} \tag{13}$$

Therefore, γ is related to $\|\mathbf{S}\|/(2L_{\mathbf{gx}})$. Since we cannot easily calculate $L_{\mathbf{gx}}$, we estimate γ by testing the difference between exact and inexact objectives during pre-computation.

To guarantee that the exact objective decreases indeed, we perform exact quasistatic simulation and evaluate the exact objective

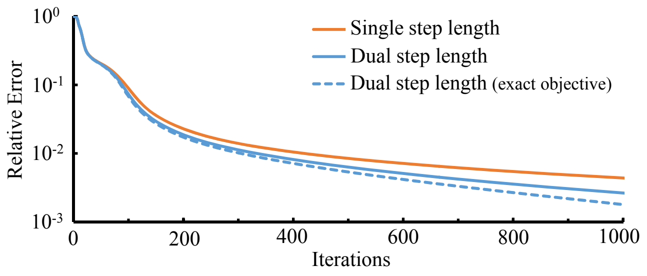


Fig. 6. The effects of step length schemes. By using a larger step length in the regions where the stiffness increases, our dual step length scheme improves the convergence rate of our inexact method. This plot also shows that exact objectives are different from inexact objectives.

in every P iterations. If the exact objective fails to decrease, we perform long-range backtracking and redo the last P iterations, with a smaller step length and more inner iterations, A and B. This practice is also known as the *watchdog strategy* [Nocedal and Wright 2006]. In our experiment, we choose the number P between 32 and 64. Long-range backtracking ensures that our inexact method converges, as it becomes the exact method in the worst case.

The aforementioned procedure works as a fail-safe to our method. If the initial step length and the numbers of inner iterations are appropriate, long-range backtracking will never occur, as shown in Fig. 6. In our experiment, we set $A = B = 256$ as our initialization by default.

A dual step length scheme. An unsurprising observation from our experiment is that the quasistatic shape changes more dramatically in the spatial regions where the stiffness decreases. This inspires us to use two step lengths: a smaller one in the regions where elastic parameters decrease; and a larger one in the regions where elastic parameters increase. In our examples, the larger one is typically 50 to 100 percent greater than the smaller one. We adjust them together when backtracking occurs. Fig. 6 shows the effectiveness of this dual step length scheme on the convergence rate.

We note that the dual step length scheme is mathematically equivalent to preconditioning the calculated gradient by a scaling matrix. Although we do not have a rigid proof yet, we believe it does not affect the conclusions given by the analysis in Subsection 4.2.

5.3 Initialization

The accuracy of the two steps also depends on the initializations of the unknowns. A natural idea is to treat their previous results as *warm starts* for the next steps. In the forward simulation step, this means $\mathbf{x}^{(l+1)}$ is initialized as: $\mathbf{x}^{(l+1)} = \mathbf{x}^{(l)} + \eta(\mathbf{x}^{(l)} - \mathbf{x}^{(l-1)})$, and in the update step, $\mathbf{a}^{(l+1)}$ is initialized as $\mathbf{a}^{(l+1)} = \eta\mathbf{a}^{(l)}$. Here η is a decay factor reducing the "temperature" of warm starts. Fig. 7 shows that the method converges slowly when $\eta = 0$ or 1, as it requires more inner iterations to achieve the accuracy requirement and avoid the divergence issue, as discussed in Subsection 4.2. The method converges fast when $\eta = 0.9$. However, it still exhibits noticeable oscillation artifacts. To avoid parameter tuning, we propose to set η as the one that minimizes the initial error. Take the linearized system in quasistatic simulation for an example: $\mathbf{J}_\mathbf{x}\left(\mathbf{x}^{(l+1)} - \mathbf{x}^{(l)}\right) = -\mathbf{f}\left(\mathbf{x}^{(l)}\right)$.

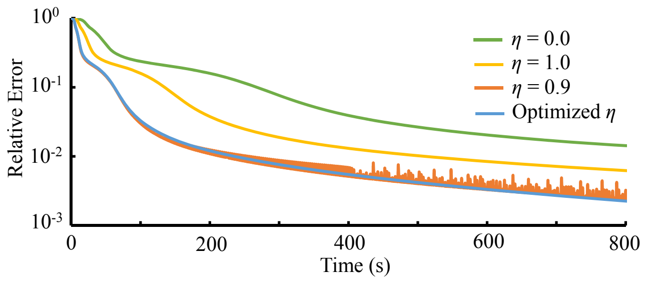


Fig. 7. The effect of the decay factor. The initializations of our fixed-point iterative solvers need a suitable decay factor, to ensure the accuracy and the convergence of the whole optimization process.

The optimal η is given by:

$$\eta = \texttt{Clamp}\left(\arg\min \left\|\eta\mathbf{J}_\mathbf{x}(\mathbf{x}^{(l)} - \mathbf{x}^{(l-1)}) + \mathbf{f}(\mathbf{x}^{(l)})\right\|^2, 0, \eta_0\right), \tag{14}$$

in which $\eta_0 = 0.9$ in our experiment. The calculation of η in Equation 14 needs one matrix-vector product and two inner products, both of which can be computed on the GPU. Fig. 7 compares the convergence rates of the method when it uses different decay factors. It shows that the optimized η leads to the most accurate results and eliminates the oscillation issue.

5.4 Regularization

The regularization term in our method contains three components for different purposes: $R(\mathbf{k}) = R_1(\mathbf{k}) + R_2(\mathbf{k}) + R_3(\mathbf{k})$. The first component, $R_1(\mathbf{k})$, is used to make the problem well-posed:

$$R_1(\mathbf{k}) = \tau_1 \|\mathbf{k}\|^2, \tag{15}$$

where τ_1 is a regularization strength constant. The second component, $R_2(\mathbf{k})$, prevents parameters from being too small or large:

$$R_2(\mathbf{k}) = \tau_2 \sum_i \left|\text{Clamp}\left(k_i, k_{\min}, k_{\max}\right) - k_i\right|^2, \tag{16}$$

where τ_2 is its strength constant and $[k_{\min}, k_{\max}]$ is the range of parameter values. Finally, the third component, $R_3(\mathbf{k})$, is used to remove high-frequency noise caused by overfitting. When each tetrahedron has its own parameters, we formulate $R_3(\mathbf{k})$ as:

$$R_3(\mathbf{k}) = \tau_3 \sum_{\{i,j\}} \frac{V_i + V_j}{2} \left(k_i - k_j\right)^2, \tag{17}$$

in which τ_3 is its strength constant, i and j are two adjacent tetrahedra, and V_i and V_j are their rest volumes. To remove the overfitting issue, τ_3 must be sufficiently large. On the other hand, if τ_3 is too large, it can cause overly smoothing artifacts. Our solution is to start with a large τ_3 first, and then gradually reduce τ_3 , until the method converges with a small τ_3.

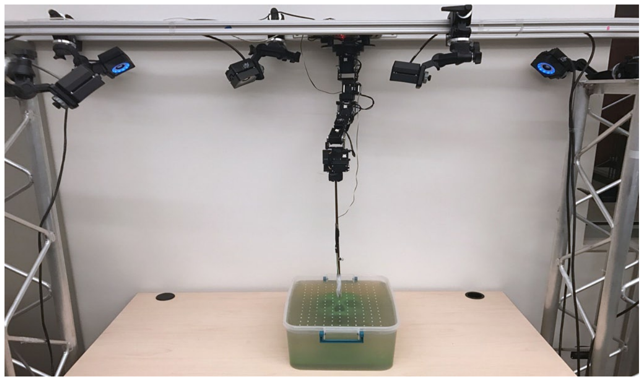
5.5 Multiple Data Samples

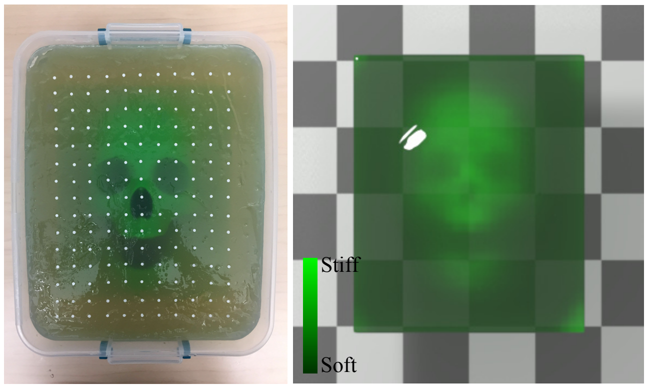
When there are multiple data samples, we formulate the constrained optimization problem as:

$$\mathbf{k} = \arg\min_{\mathbf{k}} \frac{1}{2}\left\{\sum_j \left\|\mathbf{S}\mathbf{x}^{[j]} - \mathbf{y}^{[j]}\right\|^2 + R(\mathbf{k})\right\}, \quad \text{subject to } \mathbf{f}^{[j]}(\mathbf{k}, \mathbf{x}^{[j]}) = \mathbf{0}, \tag{18}$$

Table 2. Statistics and timings. The number of tetrahedra, the number of data samples, and the number of iterations are the three major factors determining the overall computational cost of an inexact descent method.

Name	Data Type	# of Tetrahedra	# of Samples	# of Markers	Parameter Type	Elastic Model	Cost per Iteration	# of Iterations	Total Cost
Teddy bear	Synthetic	49K	20	8,317	Shear	neo-Hookean	0.71s	800	568.6s
Skull (3D)	Synthetic	130K	168	24,939	Shear	neo-Hookean	15.12s	600	9,072.5s
Skull (flat)	Captured	52K	75	165	Shear	neo-Hookean	2.21s	500	1,105.2s
Cube	Synthetic	162K	3	5,401	Bulk	neo-Hookean	0.32s	700	224.2s
Bridge	User-specified	79K	1	3	1st Shear	Mooney-Rivlin	0.12s	900	108.3s
Armadillo	User-specified	55K	1	3	Shear	neo-Hookean	0.05s	Interactive	Interactive


(a) The data capture device



(b) The 3D object (c) The result

Fig. 8. The flat skull example. Our method is able to detect the buried skull from the captured contact data. We use similar experimental settings to generate our synthetic data sets.

where $\mathbf{x}^{[j]}$ and $\mathbf{y}^{[j]}$ are the quasistatic vertex state and the observation of sample j, and $\mathbf{f}^{[j]}$ is the force function of sample j. Let there be J data samples in total. Equation 18 needs J quasistatic vertex vectors and it is subject to J quasistatic constraints specified by the force functions. Like before, our method iteratively solves a forward simulation step and an update step. The use of multiple data samples does not affect the validity of our convergence analysis in Section 4.2. It should be noted that both the quasistatic vertex state

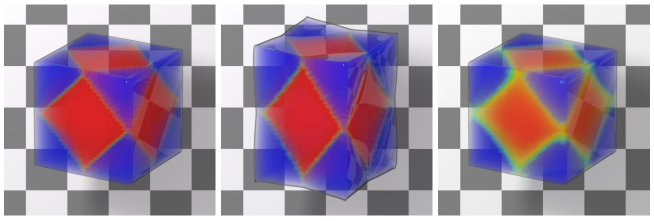

(a) The ground truth (b) The stretched shape (c) The result

Fig. 9. The cube example. The high bulk moduli in the middle of this cube causes the middle region to shrink more severely under stretching. By using data samples to capture this effect, our descent method recovers the heterogeneous bulk moduli as shown in (c).

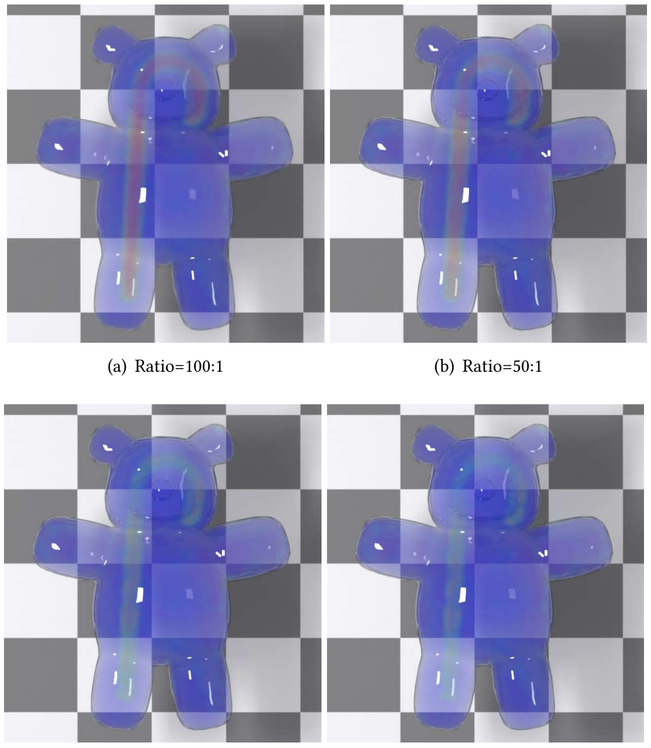
and the adjoint state can be computed in parallel for multiple samples. This allows another level of parallelization, if computational resources permit.

6 RESULTS

We use the Intel MKL PARDISO library for CPU computation and the CUDA library for GPU computation. Our experiment runs on an Intel Core i7-5930K 3.5GHz processor and an NVIDIA GeForce GTX TITAN X graphics card. The total running time of each example varies from seconds to hours, depending on the number of tetrahedra, the number of data samples, and the number of iterations as summarized in Table 2. We assign each tetrahedron with its own elastic parameters in our experiments.

Elastic material measurement. An important application of elastic parameter optimization is to measure elastic material properties of heterogeneous solids, especially when the heterogeneity is buried underneath the surface. The basic setup of our experiment is shown in Fig. 8a. A rigid skull is buried underneath soft elastic gels. A force probe makes contacts with the surface and our in-house data capture device acquires the locations of the surface markers (in white). Our synthetic data sets are acquired under similar settings, except that we can flexibly add more markers inside of the object, such as the 3D skull shown in Fig. 1.

In our experiment, we test two hyperelastic models: a compressible neo-Hookean model [Ogden 1997] and a compressible Mooney-Rivlin model [Macosko 1994]. Fig. 8c shows the optimized shear moduli of the flat skull example. From the heterogeneous material pattern, we can identify the buried solid skull object. Compared with


(a) Ratio=100:1 (b) Ratio=50:1

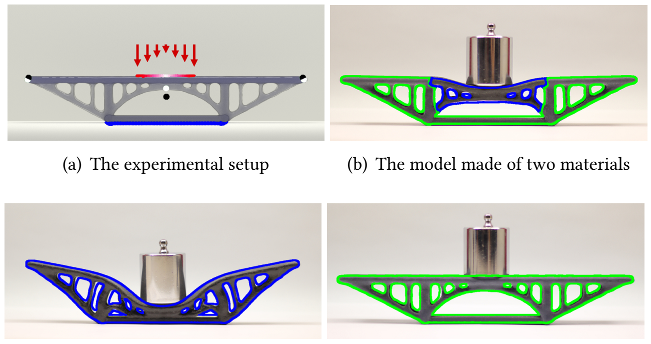
(c) Ratio=10:1 (d) Ratio=5:1

Fig. 10. The results of different shear modulus ratios. In this example, we change the ratio between the shear moduli of the two elastic materials. Our method is able to recover the heterogeneous elastic material pattern, even when the ratio is 5:1 only.

shear moduli, bulk moduli are more difficult to optimize, since their effects are less noticeable in data samples. But if data samples do capture the volume preservation effect well, our method is also able to estimate bulk moduli, as the cube example demonstrates in Fig. 9. We note that certain elastic parameters have similar behaviors and they cannot be optimized together, such as the two shear moduli under the Mooney-Rivlin model. Instead we fix one and optimize the other.

We noticed from our experiment that the optimized result tends to be smoother than the ground truth, due to limited numbers of data samples and markers. Therefore we would like to know how the result looks like, when the heterogeneous material properties are not so distinctively different. In Fig. 10, we study this issue by using four different ratios between the shear moduli of the two materials. While the result becomes blurry as this ratio decreases, we are still able to identify the solid from the heterogeneous pattern recovered by our method, as shown in Fig. 10d.

Printable material design. We use the bridge example to evaluate the usefulness of our method in designing the distribution of printable 3D materials. In this example, the bottom vertices of the bridge (in blue) are fixed to the ground floor, and the vertices on the top (in red) receive additional loads caused by a 200g calibration


(a) The experimental setup (b) The model made of two materials

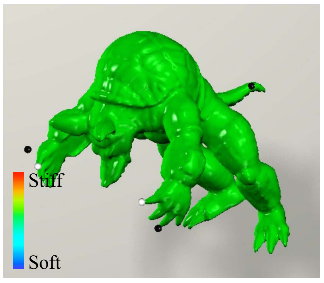
(c) The model made of Black 27A (d) The model made of Black 95A

Fig. 11. The bridge example. We estimate elastic parameters from fabricated single-material models shown in (c) and (d). We then apply our method to design the material distribution of this bridge, to achieve the deformation goals set by dark dots in (a). The contour of the simulation result in each photo illustrates its consistency with the fabricated model.

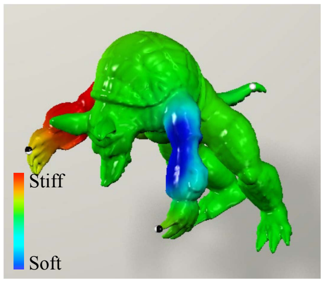
weight, as Fig. 11a shows. We distribute more forces to the sides of the weight than the center, to account for the fact that the deformed bridge floor contacts the weight mostly near the sides. We print the model using two Stratasys PolyJet materials: Black 27A with a tensile strength between 0.8MPa and 1.5MPa, and Black 95A with a tensile strength between 8.5MPa and 10MPa. Since we do not know the exact physical properties of the two materials, we print two single-material models as shown in Fig. 11c and 11d, and then apply our method to obtain the common parameters from observations. The estimated tensile strengths of Black 27A and Black 95A are 1.1MPa and 9.5MPa, respectively.

Our design goal is to lower the center of the bridge floor without affecting the two ends, which can be described as a data sample with three observed vertices (in dark), as Fig. 11a shows. Although we can formulate the design as a discrete optimization problem [Bickel et al. 2010], it is more efficient to handle it by continuous material optimization as Xu and colleagues [2015a] pointed out. The difficulty is how to convert the continuous result into a discrete material distribution without introducing too much error. Simply placing a threshold on material parameters would cause the error, i.e. the average vertex difference, to increase from 1.5mm to 5.4mm. This issue can be lessened by dithering [Xu et al. 2015a], but it will cause disconnected regions that cannot be fabricated by many printers. Thanks to the high performance of our system, we can now use an interval approach. Tetrahedra with elastic parameters outside of the interval are assigned with materials first. Otherwise, their parameters remain as unknowns. We reduce the interval and run the optimization at the same time, until all of the tetrahedra are assigned with materials. Fig. 11b shows this approach avoids disconnections and it increases the error from 1.5mm to 3.5mm only.

Interactive material design. If the number of data samples is small, our method can also be used for interactive material design of elastic objects, as shown in Fig. 12. In this example, we allow the user to specify desired finger tip heights of an armadillo model, and we run our method to achieve these goals through elastic parameter

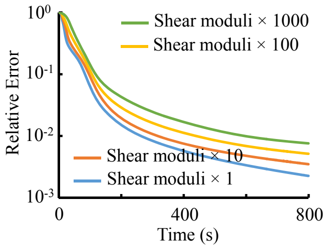


(a) Before optimization

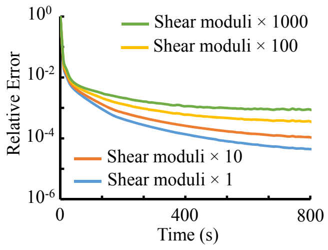


(b) After optimization

Fig. 12. The armadillo example. In this interactive example, the user specifies the desired heights of the finger tips (in black), and the method optimizes the shear modulus of every tetrahedron to reach these goals in the quasistatic state, as shown in (b).



(a) The performance of our method



(b) The performance of our simulator

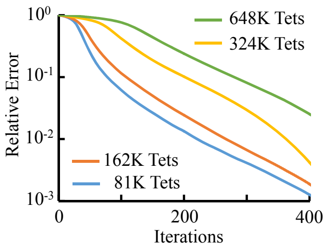
Fig. 13. The performance of our method and our forward simulator at different stiffness levels. The performance drop of our simulator is one of the main reasons for the performance drop of our method, as the shear moduli increase.

optimization on the fly. Fig. 12b shows that the method decreases the shear moduli of the left arm, while increases the shear moduli of the right arm. Doing this lowers the left arm and raises the right arm in the quasistatic shape, as expected.

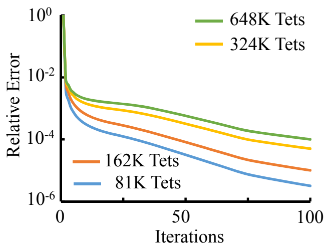
Scalability analysis. The performance of the quasistatic simulator [Wang and Yang 2016] drops as the material stiffness increases, as Fig. 13b shows. Therefore, the performance of our method also decreases as the shear moduli increase, as Fig. 13a shows.

Besides the material stiffness, the mesh resolution is another critical factor to our system performance. Here we test the scalability of our method with respect to the mesh resolution in the cube example. To prevent the convergence rate from being affected by the ratio of markers to vertices, we maintain the ratio in this experiment by using more markers at higher resolutions. Fig. 14a shows that the convergence rate of our method drops as the number of tetrahedra increases. This is consistent with the convergence rate drop of our simulator shown in Fig. 14b.

The aforementioned analysis implies that the scalability of our method is mostly determined by our forward simulator, rather than by our inexact descent optimization strategy. As quasistatic simulators become more scalable to the material stiffness and the mesh resolution, we will investigate their uses in the development of fast inexact descent methods for solving high stiffness and high resolution problems in the future.



(a) The convergence of our method



(b) The convergence of our simulator

Fig. 14. The convergence of our method and our simulator at different mesh resolutions. The convergence rate of our method decreases as the mesh resolution increases. This behavior is consistent with that of our simulator.

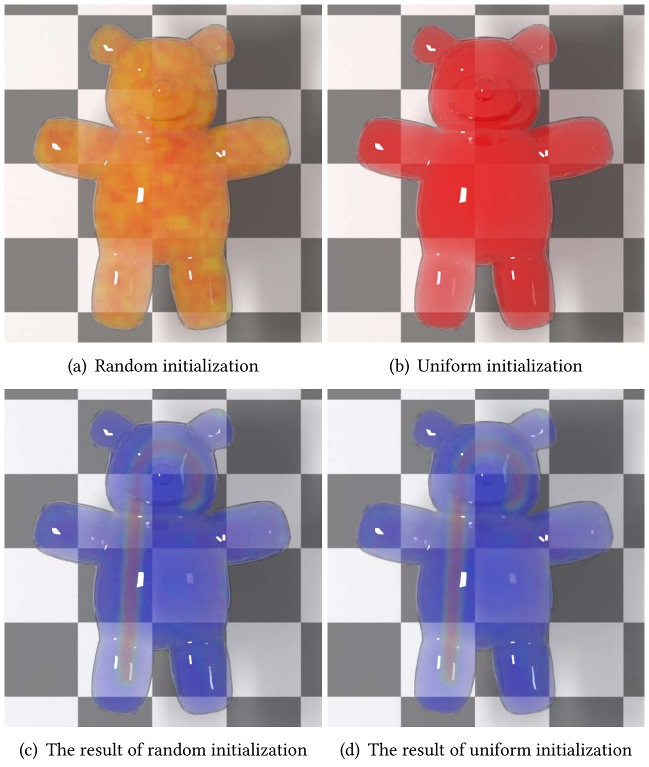
The outcome of parameter initialization. As our method potentially suffers from the local minimum issue, we want to know how bad this issue is and how sensitive the result is to parameter initialization. To answer these questions, we run the same optimization multiple times with different initial material distributions, as shown in Fig. 15a and 15b. Fig. 15c and 15d reveal that the method produces nearly identical results. We also notice that the remaining difference is mostly caused by a lack of data samples. The method tends to leave parameters untouched in the regions where deformation effects are not observed. This problem can be resolved once we use more data samples.

6.1 Limitations

Perhaps the biggest issue with our inexact descent method is that it cannot evaluate the objective function exactly. As a result, it requires multiple practical ways to guarantee its convergence, such as those described in Subsection 5.2. The result of our method depends on the data samples and it can become overly smoothed, if the regularization term is strong or the data samples are insufficient. Our method assumes that forces are applied at mesh vertices. This assumption can introduce errors near surfaces, unless the mesh is adaptively constructed. The effectiveness of our method is partly due to the choice of a GPU-based iterative forward simulator. But this causes the method to inevitably inherit many limitations of the simulator, such as the scalability to the stiffness and the dependency on the mesh quality. Currently, the method requires the reference shape of the object to be known. This requirement is often difficult to meet in real-world measurement cases, for example, when the object is in self occlusion or when it sags under gravity. Finally, a real-world object can exhibit viscoelastic, plastic, or frictional behaviors, none of which has been considered by our method yet.

7 CONCLUSIONS AND FUTURE WORK

In this paper, we propose to incorporate the inexact idea into the development of a descent method for elastic parameter optimization. Being one of its kind, our method still shares many similarities with existing inexact methods, especially inexact proximal point methods. Therefore, when the errors are small, it is not surprising to see that the method is guaranteed to converge, and it runs substantially faster than exact descent methods.


(a) Random initialization (b) Uniform initialization

(c) The result of random initialization (d) The result of uniform initialization

Fig. 15. The results of different initializations. Our method produces nearly identical elastic parameter patterns as shown in (c) and (d), regardless of their initializations in (a) and (b).

In the near future, we would like to improve the performance of our method by multiple GPUs and multigrid techniques. We then plan to investigate the use of our method in elastic parameter optimization with frictional contacts, in which case unilateral constraints exist. In the long term, we are interested in developing inexact methods for designing and measuring other solid material properties, such as viscoelasticity and plasticity. Finally, we would like to apply the inexact idea to solve more inverse elastic problems, such as elastic shape design and space-time optimization.

ACKNOWLEDGMENTS

This work was funded by NSF grant CHS-1524992 and NSFC grant 61332017. The authors would also like to thank Adobe and NVIDIA for additional equipment and funding supports.

REFERENCES

David Baraff and Andrew Witkin. 1998. Large Steps in Cloth Simulation. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques (SIGGRAPH '98)*. ACM, New York, NY, USA, 43–54.

Markus Becker and Matthias Teschner. 2007. Robust and Efficient Estimation of Elasticity Parameters Using the Linear Finite Element Method. In *SimVis*. 15–28.

Luca Bergamaschi, Igor Moret, and Giovanni Zilli. 2001. Inexact Quasi-Newton Methods for Sparse Systems of Nonlinear Equations. *Future Generation Computer Systems* 18, 1 (Sept. 2001), 41–53.

Hans-Uwe Berger. 2009. *Inverse Problems in Soft Tissue Elastography using Boundary Element Methods.* Ph.D. Dissertation. University of Canterbury.

Kiran S. Bhat, Christopher D. Twigg, Jessica K. Hodgins, Pradeep K. Khosla, Zoran Popović, and Steven M. Seitz. 2003. Estimating Cloth Simulation Parameters from Video. In *Proceedings of SCA*. 37–51.

Bernd Bickel, Moritz Bächer, Miguel A. Otaduy, Hyunho Richard Lee, Hanspeter Pfister, Markus Gross, and Wojciech Matusik. 2010. Design and Fabrication of Materials with Desired Deformation Behavior. *ACM Trans. Graph. (SIGGRAPH)* 29, 4, Article 63 (July 2010), 10 pages.

Bernd Bickel, Moritz Bächer, Miguel A. Otaduy, Wojciech Matusik, Hanspeter Pfister, and Markus Gross. 2009. Capture and Modeling of Non-linear Heterogeneous Soft Tissue. *ACM Trans. Graph. (SIGGRAPH)* 28, 3, Article 89 (July 2009), 9 pages.

Ernesto G. Birgin, Nataša Krejić, and José Mario Martínez. 2003. Globally Convergent Inexact Quasi-Newton Methods for Solving Nonlinear Systems. *Numerical Algorithms* 32, 2 (April 2003), 249–260.

Marc Bonnet and Andrei Constantinescu. 2005. Inverse Problems in Elasticity. *Inverse Problems* 21, 2 (2005), 1–50.

Sofien Bouaziz, Sebastian Martin, Tiantian Liu, Ladislav Kavan, and Mark Pauly. 2014. Projective Dynamics: Fusing Constraint Projections for Fast Simulation. *ACM Trans. Graph. (SIGGRAPH)* 33, 4, Article 154 (July 2014), 11 pages.

Regina Burachik and Joydeep Dutta. 2010. Inexact Proximal Point Methods for Variational Inequality Problems. *SIAM Journal on Optimization* 20, 5 (2010), 2653–2678.

Richard H. Byrd, Frank E. Curtis, and Jorge Nocedal. 2008. An Inexact SQP Method for Equality Constrained Optimization. *SIAM Journal on Optimization* 19, 1 (2008), 351–369.

Romain Casati, Gilles Daviet, and Florence Bertails-Descoubes. 2016. *Inverse Elastic Cloth Design with Contact and Friction.* Technical report. Inria Grenoble Rhône-Alpes, Université de Grenoble.

Xiang Chen, Changxi Zheng, Weiwei Xu, and Kun Zhou. 2014. An Asymptotic Numerical Method for Inverse Elastic Shape Design. *ACM Trans. Graph. (SIGGRAPH)* 33, 4, Article 95 (July 2014), 11 pages.

Ron S. Dembo, Stanley C. Eisenstat, and Trond Steihaug. 1982. Inexact Newton Methods. *SIAM J. Numer. Anal.* 19, 2 (April 1982), 400–408.

Marvin M. Doyley. 2012. Model-Based Elastography: A Survey of Approaches to the Inverse Elasticity Problem. *Physics in Medicine and Biology* 57, 3 (2012), 35–73.

Stanley C. Eisenstat and Homer F. Walker. 1994. Globally Convergent Inexact Newton Methods. *SIAM Journal on Optimization* 4, 2 (1994), 393–422.

Marco Fratarcangeli, Valentina Tibaldo, and Fabio Pellacini. 2016. Vivace: A Practical Gauss-Seidel Method for Stable Soft Body Dynamics. *ACM Trans. Graph. (SIGGRAPH Asia)* 35, 6, Article 214 (Nov. 2016), 9 pages.

Mark S. Gockenbach, Baasansuren Jadamba, Akhtar A. Khan, Christiane Tammer, and Brian Winkler. 2015. Proximal Methods for the Elastography Inverse Problem of Tumor Identification Using an Equation Error Approach. In *Advances in Variational and Hemivariational Inequalities*. Springer, Chapter 10, 173–197.

Gene H. Golub and Charles F. Van Loan. 2012. *Matrix Computations (4th Ed.).* Johns Hopkins University Press, Baltimore, MD, USA.

Eldad Haber, Uri M. Ascher, and Douglas W. Oldenburg. 2004. Inversion of 3D Electromagnetic Data in Frequency and Time Domain Using an Inexact All-At-Once Approach. *Geophysics* 69, 5 (2004), 1216–1228.

Tyler S. Kaster, Ira Sack, and Afshan Samani. 2011. Measurement of the Hyperelastic Properties of *ex vivo* Brain Tissue Slices. *Journal of Biomechanics* 44, 6 (April 2011), 1158–1163.

Mahmud Khodadad and Mohsen Dashti Ardakani. 2009. Application of the Inverse Elasticity Problem to Identify Irregular Interfacial Configurations. *Engineering Analysis with Boundary Elements* 33, 6 (June 2009), 872–879.

Tiantian Liu, Adam W. Bargteil, James F. O'Brien, and Ladislav Kavan. 2013. Fast Simulation of Mass-Spring Systems. *ACM Trans. Graph. (SIGGRAPH Asia)* 32, 6, Article 214 (Nov. 2013), 7 pages.

Zohreh Barani Lonbani. 2010. *Elastographic Reconstruction Methods for Orthotropic Materials.* Master's thesis. University of Canterbury.

Erik Lund, Henrik Møller, and Lars A. Jakobsen. 2003. Shape Design Optimization of Stationary Fluid-Structure Interaction Problems with Large Displacements and Turbulence. *Structural and Multidisciplinary Optimization* 25, 5–6 (Dec. 2003), 383–392.

Christopher W. Macosko. 1994. *Rheology: Principles, Measurement and Applications.* VCH Publishers.

Nicholas Maratos. 1978. *Exact Penalty Function Algorithms for Finite-Dimensional and Control Optimization Problems.* Ph.D. Dissertation. University of London.

Vittorio Megaro, Jonas Zehnder, Moritz Bächer, Stelian Coros, Markus Gross, and Bernhard Thomaszewski. 2017. A Computational Design Tool for Compliant Mechanisms. *ACM Trans. Graph. (SIGGRAPH)* 36, 4, Article 82 (July 2017), 12 pages.

Eder Miguel, Derek Bradley, Bernhard Thomaszewski, Bernd Bickel, Wojciech Matusik, Miguel A. Otaduy, and Steve Marschner. 2012. Data-Driven Estimation of Cloth Simulation Models. *Comput. Graph. Forum (Eurographics)* 31, 2 (May 2012), 519–528.

Eder Miguel, David Miraut, and Miguel A. Otaduy. 2016. Modeling and Estimation of Energy-Based Hyperelastic Objects. *Computer Graphics Forum (Eurographics)* 35, 2 (May 2016), 385–396.

Matthias Müller. 2008. Hierarchical Position Based Dynamics. In *Proceedings of VRIPHYS*. 1–10.

Matthias Müller, Bruno Heidelberger, Matthias Teschner, and Markus Gross. 2005. Meshless Deformations Based on Shape Matching. *ACM Trans. Graph. (SIGGRAPH)* 24, 3 (July 2005), 471–478.

Rahul Narain, Matthew Overby, and George E. Brown. 2016. ADMM ⊇ Projective Dynamics: Fast Simulation of General Constitutive Models. In *Proceedings of SCA*. 21–28.

Jorge Nocedal and Stephen J. Wright. 2006. *Numerical Optimization (2rd Ed.)*. Springer.

James F. O'Brien and Jessica K. Hodgins. 1999. Graphical Modeling and Animation of Brittle Fracture. In *Proceedings of SIGGRAPH 98 (Annual Conference Series)*. 137–146.

Ray W. Ogden. 1997. *Non-linear Elastic Deformations*. Dover Publications, Inc.

Joseph J. O'Hagan and Afshan Samani. 2008. Measurement of the Hyperelastic Properties of Tissue Slices with Tumour Inclusion. *Physics in Medicine and Biology* 53, 24 (Dec. 2008), 7087–7106.

Dinesh K. Pai, Kees van den Doel, Doug L. James, Jochen Lang, John E. Lloyd, Joshua L. Richmond, and Som H. Yau. 2001. Scanning Physical Interaction Behavior of 3D Objects. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '01)*. 87–96.

Roger P. Pawlowski, Joseph P. Simonis, Homer F. Walker, and John N. Shadid. 2008. Inexact Newton Dogleg Methods. *SIAM J. Numer. Anal.* 46, 4 (May 2008), 2112–2132.

Jesús Pérez, Miguel A. Otaduy, and Bernhard Thomaszewski. 2017. Computational Design and Automated Fabrication of Kirchhoff-Plateau Surfaces. *ACM Trans. Graph. (SIGGRAPH)* 36, 4, Article 62 (July 2017), 12 pages.

Jesús Pérez, Bernhard Thomaszewski, Stelian Coros, Bernd Bickel, José A. Canabal, Robert Sumner, and Miguel A. Otaduy. 2015. Design and Fabrication of Flexible Rod Meshes. *ACM Trans. Graph. (SIGGRAPH)* 34, 4, Article 138 (July 2015), 12 pages.

Xavier Provot. 1996. Deformation Constraints in a Mass-Spring Model to Describe Rigid Cloth Behavior. In *Proceedings of Graphics Interface*. 147–154.

Rien Quirynen, Sebastien Gros, and Moritz Diehl. 2017. Inexact Newton-Type Optimization with Iterated Sensitivities. *SIAM Journal on Optimization* 28, 1 (July 2017), 74–95.

Mikhail V. Solodov and Benar F. Svaiter. 2001. A Unified Framework for Some Inexact Proximal Point Algorithms. *Numerical Functional Analysis and Optimization* 22, 7–8 (2001), 1013–1035.

Joseph Teran, Silvia Blemker, V Ng Thow Hing, and Ron Fedkiw. 2003. Finite Volume Methods for the Simulation of Skeletal Muscle. In *Proceedings of SCA*. 68–74.

Joseph Teran, Eftychios Sifakis, Geoffrey Irving, and Ronald Fedkiw. 2005. Robust Quasistatic Finite Elements and Flesh Simulation. In *Proceedings of SCA*. 181–190.

Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. 1987. Elastically Deformable Models. *SIGGRAPH Comput. Graph.* 21, 4 (Aug. 1987), 205–214.

Bernhard Thomaszewski, Simon Pabst, and Wolfgang Strasser. 2009. Continuum-Based Strain Limiting. *Computer Graphics Forum (Eurographics)* 28, 2 (2009), 569–576.

Nobuyuki Umetani, Danny M. Kaufman, Takeo Igarashi, and Eitan Grinspun. 2011. Sensitive Couture for Interactive Garment Modeling and Editing. *ACM Trans. Graph. (SIGGRAPH)* 30, 4, Article 90 (July 2011), 12 pages.

Bin Wang, Longhua Wu, KangKang Yin, Uri Ascher, Libin Liu, and Hui Huang. 2015. Deformation Capture and Modeling of Soft Objects. *ACM Trans. Graph. (SIGGRAPH)* 34, 4, Article 94 (July 2015), 12 pages.

Huamin Wang. 2015. A Chebyshev Semi-Iterative Approach for Accelerating Projective and Position-Based Dynamics. *ACM Trans. Graph. (SIGGRAPH Asia)* 34, 6, Article 246 (Oct. 2015), 9 pages.

Huamin Wang. 2018. Rule-Free Sewing Pattern Adjustment with Precision and Efficiency. *ACM Trans. Graph. (SIGGRAPH)* 37, 4, Article 53 (July 2018), 13 pages.

Huamin Wang, James O'Brien, and Ravi Ramamoorthi. 2010. Multi-Resolution Isotropic Strain Limiting. *ACM Trans. Graph. (SIGGRAPH Asia)* 29, 6, Article 156 (Dec. 2010), 10 pages.

Huamin Wang, James F. O'Brien, and Ravi Ramamoorthi. 2011. Data-Driven Elastic Models for Cloth: Modeling and Measurement. *ACM Trans. Graph. (SIGGRAPH)* 30, 4, Article 71 (July 2011), 12 pages.

Huamin Wang and Yin Yang. 2016. Descent Methods for Elastic Body Simulation on the GPU. *ACM Trans. Graph. (SIGGRAPH Asia)* 35, 6, Article 212 (Nov. 2016), 10 pages.

Hongyi Xu, Yijing Li, Yong Chen, and Jernej Barbič. 2015a. Interactive Material Design Using Model Reduction. *ACM Trans. Graph.* 34, 2, Article 18 (March 2015), 14 pages.

Hongyi Xu, Funshing Sin, Yufeng Zhu, and Jernej Barbič. 2015b. Nonlinear Material Design Using Principal Stretches. *ACM Trans. Graph. (SIGGRAPH)* 34, 4, Article 75 (July 2015), 11 pages.

Jonas Zehnder, Espen Knoop, Moritz Bächer, and Bernhard Thomaszewski. 2017. Metasilicone: Design and Fabrication of Composite Silicone with Desired Mechanical Properties. *ACM Trans. Graph. (SIGGRAPH Asia)* 36, 6, Article 240 (Nov. 2017), 13 pages.

A CONVERGENCE ANALYSIS

LEMMA A.1. *Let $\mathcal{G}(\mathbf{k},\mathbf{x},\mathcal{A}(\mathbf{k},\mathbf{x}))$ be Lipschitz continuous with respect to $\mathbf{x}$, $\mathcal{G}(\mathbf{k},\mathbf{x},\mathbf{a})$ be Lipschitz continuous with respect to $\mathbf{a}$, $L_{\mathbf{gx}}$ and $L_{\mathbf{ga}}$ be the Lipschitz constants. If $\left\|\nabla C(\mathbf{k}^{(l)})\right\| \neq 0$, $\left\|\mathbf{x}^{(l)} - \mathcal{X}(\mathbf{k}^{(l)})\right\| < \frac{1}{2L_{\mathrm{gx}}}\left\|\nabla C(\mathbf{k}^{(l)})\right\|$ and $\left\|\mathbf{a}^{(l)} - \mathcal{A}(\mathbf{k}^{(l)},\mathbf{x}^{(l)})\right\| < \frac{1}{2L_{\mathrm{ga}}}\left\|\nabla C(\mathbf{k}^{(l)})\right\|$, then $-\mathbf{g}^{(l)}\cdot\nabla C(\mathbf{k}^{(l)}) < 0$.*

PROOF. From the given conditions, we get:

$$\begin{aligned}&\left\|\mathcal{G}(\mathbf{k}^{(l)},\mathbf{x}^{(l)},\mathcal{A}(\mathbf{k}^{(l)},\mathbf{x}^{(l)})) - \nabla C(\mathbf{k}^{(l)})\right\|\\ &\quad\leq L_{\mathbf{gx}}\left\|\mathbf{x}^{(l)} - \mathcal{X}(\mathbf{k}^{(l)})\right\| < \tfrac{1}{2}\left\|\nabla C(\mathbf{k}^{(l)})\right\|,\end{aligned}\tag{19}$$

and,

$$\begin{aligned}&\left\|\mathbf{g}^{(l)} - \mathcal{G}(\mathbf{k}^{(l)},\mathbf{x}^{(l)},\mathcal{A}(\mathbf{k}^{(l)},\mathbf{x}^{(l)}))\right\|\\ &\quad\leq L_{\mathbf{ga}}\left\|\mathbf{a}^{(l)} - \mathcal{A}(\mathbf{k}^{(l)},\mathbf{x}^{(l)})\right\| < \tfrac{1}{2}\left\|\nabla C(\mathbf{k}^{(l)})\right\|.\end{aligned}\tag{20}$$

Together, we know $\left\|\mathbf{g}^{(l)} - \nabla C(\mathbf{k}^{(l)})\right\| < \left\|\nabla C(\mathbf{k}^{(l)})\right\|$ and we have:

$$\begin{aligned}&\mathbf{g}^{(l)}\cdot\nabla C(\mathbf{k}^{(l)})\\ &\quad= \left(\nabla C(\mathbf{k}^{(l)}) - \left(-\mathbf{g}^{(l)} + \nabla C(\mathbf{k}^{(l)})\right)\right)\cdot\nabla C(\mathbf{k}^{(l)})\\ &\quad\geq \left(\left\|\nabla C(\mathbf{k}^{(l)})\right\| - \left\|\mathbf{g}^{(l)} - \nabla C(\mathbf{k}^{(l)})\right\|\right)\left\|\nabla C(\mathbf{k}^{(l)})\right\| > 0.\end{aligned}\tag{21}$$

Therefore the statement is true. □

THEOREM A.2. *Let $\mathcal{G}(\mathbf{k},\mathbf{x},\mathcal{A}(\mathbf{k},\mathbf{x}))$ be Lipschitz continuous with respect to $\mathbf{x}$, $\mathcal{G}(\mathbf{k},\mathbf{x},\mathbf{a})$ be Lipschitz continuous with respect to $\mathbf{a}$, and $\nabla C(\mathbf{k})$ and $\mathcal{X}(\mathbf{k})$ be Lipschitz continuous with respect to $\mathbf{k}$. If the error conditions are met in the l-th outer iteration, there exists a constant μ, such that $\left\|\mathbf{x}^{(l+1)} - \mathcal{X}(\mathbf{k}^{(l+1)})\right\| \leq \frac{1}{2L_{\mathrm{gx}}}\left\|\nabla C(\mathbf{k}^{(l+1)})\right\|$, if $\mathbf{x}^{(l+1)}$ satisfies $\left\|\mathbf{x}^{(l+1)} - \mathcal{X}(\mathbf{k}^{(l+1)})\right\| \leq \mu\left\|\mathbf{x}^{(l)} - \mathcal{X}(\mathbf{k}^{(l+1)})\right\|$.*

PROOF. By definition, $\mathbf{k}^{(l+1)} = \mathbf{k}^{(l)} - s^{(l)}\mathbf{g}^{(l)}$, for $s^{(l)} \in (0, S]$. Together with Lemma A.1, we have:

$$\begin{aligned}&\left\|\mathbf{x}^{(l+1)} - \mathcal{X}(\mathbf{k}^{(l+1)})\right\|\\ &\quad\leq \mu\left(\left\|\mathbf{x}^{(l)} - \mathcal{X}(\mathbf{k}^{(l)})\right\| + \left\|\mathcal{X}(\mathbf{k}^{(l+1)}) - \mathcal{X}(\mathbf{k}^{(l)})\right\|\right)\\ &\quad\leq \mu\left(\tfrac{1}{2L_{\mathrm{gx}}}\left\|\nabla C(\mathbf{k}^{(l)})\right\| + s^{(l)}L_{\mathbf{x}}\left\|\mathbf{g}^{(l)}\right\|\right)\\ &\quad\leq \mu\left(\tfrac{1}{2L_{\mathrm{gx}}} + 2SL_{\mathbf{x}}\right)\left\|\nabla C(\mathbf{k}^{(l)})\right\|,\end{aligned}\tag{22}$$

in which $L_{\mathbf{x}}$ is the Lipschitz constant of $\mathcal{X}(\mathbf{k})$. Meanwhile, we build a relationship between $\left\|\nabla C(\mathbf{k}^{(l)})\right\|$ and $\left\|\nabla C(\mathbf{k}^{(l+1)})\right\|$:

$$\begin{aligned}\left\|\nabla C(\mathbf{k}^{(l+1)})\right\| &\geq \left\|\nabla C(\mathbf{k}^{(l)})\right\| - \left\|\nabla C(\mathbf{k}^{(l+1)}) - \nabla C(\mathbf{k}^{(l)})\right\|\\ &\geq \left\|\nabla C(\mathbf{k}^{(l)})\right\| - L_{\mathbf{gk}}\left\|\mathbf{k}^{(l+1)} - \mathbf{k}^{(l)}\right\|\\ &\geq \left\|\nabla C(\mathbf{k}^{(l)})\right\| - s^{(l)}L_{\mathbf{gk}}\left\|\mathbf{g}^{(l)}\right\|\\ &\geq (1 - 2SL_{\mathbf{gk}})\left\|\nabla C(\mathbf{k}^{(l)})\right\|.\end{aligned}\tag{23}$$

in which $L_{\mathbf{gk}}$ is the Lipschitz constant of $\nabla C(\mathbf{k})$. From Equation 22 and 23, we get:

$$\left\|\mathbf{x}^{(l+1)} - \mathcal{X}(\mathbf{k}^{(l+1)})\right\| \leq \mu\left(\frac{\frac{1}{2L_{\mathrm{gx}}} + 2SL_{\mathrm{x}}}{1 - 2SL_{\mathrm{gk}}}\right)\left\|\nabla C(\mathbf{k}^{(l+1)})\right\|.\tag{24}$$

Therefore, the first error condition is met in the l+1-th iteration, if $0 \le \mu \le \frac{1-2SL_{\mathrm{gk}}}{1+4SL_{\mathrm{x}}L_{\mathrm{gx}}}$. □

Theorem A.3. *Let $\mathcal{G}(\mathbf{k},\mathbf{x},\mathcal{A}(\mathbf{k},\mathbf{x}))$ be Lipschitz continuous with respect to $\mathbf{x}$, $\mathcal{G}(\mathbf{k},\mathbf{x},\mathbf{a})$ be Lipschitz continuous with respect to $\mathbf{a}$, $\nabla C(\mathbf{k})$, $\mathcal{X}(\mathbf{k})$ and $\mathcal{A}(\mathbf{k},\mathbf{x})$ be Lipschitz continuous as well. If the error conditions are met in the l-th outer iteration and the first error condition is met in the next outer iteration, there exists a constant ξ, such that $\left\|\mathbf{a}^{(l+1)}-\mathcal{A}(\mathbf{k}^{(l+1)},\mathbf{x}^{(l+1)})\right\| \le \frac{1}{2L_{\mathrm{ga}}}\left\|\nabla C(\mathbf{k}^{(l+1)})\right\|$, if $\mathbf{a}^{(l+1)}$ satisfies $\left\|\mathbf{a}^{(l+1)}-\mathcal{A}(\mathbf{k}^{(l+1)},\mathbf{x}^{(l+1)})\right\| \le \xi\left\|\mathbf{a}^{(l)}-\mathcal{A}(\mathbf{k}^{(l+1)},\mathbf{x}^{(l+1)})\right\|$.*

Proof. By definition, $\mathbf{k}^{(l+1)} = \mathbf{k}^{(l)} - s^{(l)}\mathbf{g}^{(l)}$, for $s^{(l)} \in (0,S]$. Together with Lemma A.1, we have:

$$\begin{aligned}&\left\|\mathbf{a}^{(l+1)}-\mathcal{A}(\mathbf{k}^{(l+1)},\mathbf{x}^{(l+1)})\right\|\\ &\le \xi\left\|\mathbf{a}^{(l)}-\mathcal{A}(\mathbf{k}^{(l+1)},\mathbf{x}^{(l+1)})\right\|\\ &\le \xi\Big(\left\|\mathbf{a}^{(l)}-\mathcal{A}(\mathbf{k}^{(l)},\mathbf{x}^{(l)})\right\|+\left\|\mathcal{A}(\mathbf{k}^{(l+1)},\mathbf{x}^{(l)})-\mathcal{A}(\mathbf{k}^{(l)},\mathbf{x}^{(l)})\right\|\\ &\qquad+\left\|\mathcal{A}(\mathbf{k}^{(l+1)},\mathbf{x}^{(l+1)})-\mathcal{A}(\mathbf{k}^{(l+1)},\mathbf{x}^{(l)})\right\|\Big)\\ &\le \xi\left(\tfrac{1}{2L_{\mathrm{ga}}}\left\|\nabla C(\mathbf{k}^{(l)})\right\|+SL_{\mathbf{ak}}\left\|\mathbf{g}^{(l)}\right\|+L_{\mathbf{ax}}\left\|\mathbf{x}^{(l+1)}-\mathbf{x}^{(l)}\right\|\right),\end{aligned} \tag{25}$$

in which $L_{\mathbf{ak}}$ and $L_{\mathbf{ax}}$ are the two Lipschitz constants of $\mathcal{A}(\mathbf{k},\mathbf{x})$. From Theorem A.2, we know:

$$\begin{aligned}&\left\|\mathbf{x}^{(l+1)}-\mathbf{x}^{(l)}\right\|\\ &\le \left\|\mathbf{x}^{(l)}-\mathcal{X}(\mathbf{k}^{(l)})\right\|+\left\|\mathbf{x}^{(l+1)}-\mathcal{X}(\mathbf{k}^{(l+1)})\right\|+\left\|\mathcal{X}(\mathbf{k}^{(l+1)})-\mathcal{X}(\mathbf{k}^{(l)})\right\|\\ &\le \tfrac{1}{2L_{\mathrm{gx}}}\left(\left\|\nabla C(\mathbf{k}^{(l)})\right\|+\left\|\nabla C(\mathbf{k}^{(l+1)})\right\|\right)+SL_{\mathbf{x}}\left\|\mathbf{g}^{(l)}\right\|.\end{aligned} \tag{26}$$

Combining Equation 23, 25, and 26, we obtain:

$$\begin{aligned}&\left\|\mathbf{a}^{(l+1)}-\mathcal{A}(\mathbf{k}^{(l+1)},\mathbf{x}^{(l+1)})\right\|\\ &\le \xi\left(\frac{\frac{1}{2L_{\mathrm{ga}}}+\frac{L_{\mathrm{ax}}}{2L_{\mathrm{gx}}}+2S(L_{\mathrm{ak}}+L_{\mathrm{ax}}L_{\mathrm{x}})}{1-2SL_{\mathrm{gk}}}+\frac{L_{\mathrm{ax}}}{2L_{\mathrm{gx}}}\right)\left\|\nabla C(\mathbf{k}^{(l+1)})\right\|.\end{aligned} \tag{27}$$

Therefore, the second error condition is met in the l+1-th iteration, if $0 \le \xi \le \frac{(1-2SL_{\mathrm{gk}})L_{\mathrm{gx}}}{L_{\mathrm{gx}}+4S(L_{\mathrm{ak}}+L_{\mathrm{ax}}L_{\mathrm{x}})L_{\mathrm{gx}}L_{\mathrm{ga}}+L_{\mathrm{ax}}(2-2SL_{\mathrm{gk}})L_{\mathrm{ga}}}$. □

B LINEAR CONVERGENCE

Suppose that $\nabla^2 C(\mathbf{k}^*)$ is positive definite and $m^*\mathbf{I} \le \nabla^2 C(\mathbf{k}^*) \le M^*\mathbf{I}$, in which $\mathbf{k}^*$ is the exact solution. If the line search is exact, an inexact descent method, fundamentally as a line search method, is globally linear convergent ([Nocedal and Wright 2006], Theorem 3.4):

$$C(\mathbf{k}^{(l+1)})-C(\mathbf{k}^*) \le \left(\frac{M^*-m^*}{M^*+m^*}\right)^2\left(C(\mathbf{k}^{(l)})-C(\mathbf{k}^*)\right). \tag{28}$$

Unfortunately, this conclusion cannot be extended to backtracking line search, due to saddle point traps and other issues. Instead, we focus our analysis on the local convergence rate of the method within a sufficiently small neighborhood of $\mathbf{k}^*$. For any $\mathbf{k}$ in this region, there exists $m > 0$ and $M > 0$, such that $m\mathbf{I} \le \nabla^2 C(\mathbf{k}) \le M\mathbf{I}$.

In addition, we assume that the errors of the two steps are smaller:

$$\begin{aligned}\left\|\mathbf{x}^{(l)}-\mathcal{X}(\mathbf{k}^{(l)})\right\| &< \tfrac{1}{4L_{\mathrm{gx}}}\left\|\nabla C(\mathbf{k}^{(l)})\right\|,\\ \left\|\mathbf{a}^{(l)}-\mathcal{A}(\mathbf{k}^{(l)},\mathbf{x}^{(l)})\right\| &< \tfrac{1}{4L_{\mathrm{ga}}}\left\|\nabla C(\mathbf{k}^{(l)})\right\|,\end{aligned} \tag{29}$$

so the error of the gradient is smaller as well:

$$\left\|\mathbf{g}^{(l)}-\nabla C(\mathbf{k}^{(l)})\right\| < \frac{1}{2}\left\|\nabla C(\mathbf{k}^{(l)})\right\|, \tag{30}$$

and we have:

$$\frac{1}{2}\left\|\nabla C(\mathbf{k}^{(k)})\right\|^2 < \nabla C(\mathbf{k}^{(l)})\cdot\mathbf{g}^{(l)} < \frac{3}{2}\left\|\nabla C(\mathbf{k}^{(k)})\right\|^2. \tag{31}$$

We note that theorem A.2 and A.3 are still valid, but with different variable values.

First, we derive a lower bound on the step length $s^{(l)}$, based on the termination condition of Backtracking-Armijo line search.

Lemma B.1. *Let $\alpha \in (0,1/6]$ be the constant for the Armijo condition: $C(\mathbf{k}^{(l)}-s^{(l)}\mathbf{g}^{(l)}) < C(\mathbf{k}^{(l)}) - \alpha s^{(l)}\mathbf{g}^{(l)}\cdot\nabla C(\mathbf{k}^{(l)})$, and $\beta \in (0,1)$ be the decrease constant: $s^{(l)} \leftarrow \beta s^{(l)}$. If $s^{(l)} \in (0,S]$, then $s^{(l)} \ge \min\left(S,\frac{2\beta}{9M}\right)$ after termination.*

Proof. Using the definition of M, Equation 30 and 31, we know:

$$\begin{aligned}&C(\mathbf{k}^{(l)}-s^{(l)}\mathbf{g}^{(l)})\\ &\le C(\mathbf{k}^{(l)})-s^{(l)}\nabla C(\mathbf{k}^{(l)})\cdot\mathbf{g}^{(l)}+\frac{M\left(s^{(l)}\right)^2}{2}\left\|\mathbf{g}^{(l)}\right\|^2\\ &< C(\mathbf{k}^{(l)})+\left(\frac{9M\left(s^{(l)}\right)^2}{8}-\frac{s^{(l)}}{2}\right)\left\|\nabla C(\mathbf{k}^{(l)})\right\|^2\\ &\le C(\mathbf{k}^{(l)})-6\alpha\frac{s^{(l)}}{4}\left\|\nabla C(\mathbf{k}^{(l)})\right\|^2\\ &< C(\mathbf{k}^{(l)})-\alpha s^{(l)}\mathbf{g}^{(l)}\cdot\nabla C(\mathbf{k}^{(l)}),\end{aligned} \tag{32}$$

if $s^{(l)} \in \left(0,\frac{2}{9M}\right]$. Therefore, backtracking line search terminates with either $s^{(l)} = S$, or $s^{(l)} \ge \frac{2\beta}{9M}$. □

Next we derive the local convergence rate.

Theorem B.2. *Suppose that all of the previous assumptions are satisfied. An inexact descent method has a linear convergence rate, if $\mathbf{k}^{(l)}$ is sufficiently close to $\mathbf{k}^*$.*

Proof. Using the definition of m and Equation 31, we get:

$$\begin{aligned}C(\mathbf{k}^*) &\ge C(\mathbf{k}^{(l)})+\nabla C(\mathbf{k}^{(l)})\cdot(\mathbf{k}^*-\mathbf{k}^{(l)})+\tfrac{m}{2}\left\|\mathbf{k}^*-\mathbf{k}^{(l)}\right\|^2\\ &= C(\mathbf{k}^{(l)})+\tfrac{m}{2}\left\|\mathbf{k}^*-\mathbf{k}^{(l)}+\tfrac{1}{m}\nabla C(\mathbf{k}^{(l)})\right\|^2-\tfrac{1}{2m}\left\|\nabla C(\mathbf{k}^{(l)})\right\|^2\\ &> C(\mathbf{k}^{(l)})-\tfrac{1}{m}\mathbf{g}^{(l)}\cdot\nabla C(\mathbf{k}^{(l)}),\end{aligned} \tag{33}$$

Together with Lemma B.1, we obtain:

$$\begin{aligned}C(\mathbf{k}^{(l+1)})-C(\mathbf{k}^*) &\le C(\mathbf{k}^{(l)})-C(\mathbf{k}^*)-\alpha s^{(l)}\mathbf{g}^{(l)}\cdot\nabla C(\mathbf{k}^{(l)})\\ &\le \left(1-\min\left(\alpha m S,\tfrac{2\alpha\beta m}{9M}\right)\right)\left(C(\mathbf{k}^{(l)})-C(\mathbf{k}^*)\right),\end{aligned} \tag{34}$$

which indicates the local convergence rate is linear. □