Numerical Methods for Linear Complementarity Problems in Physics-based Animation

Kenny Erleben
Department of Computer Science
University of Copenhagen
kenny@diku.dk

February 2013

Abstract

In physics-based animation linear complementarity problems (LCPs) have historically been used as models of contact forces between rigid bodies. Recently LCPs are being deployed for other types of animation like deformable models, fluids, and granular material. Thus, LCPs are becoming a general important fundamental model. Hence, there is a real need for providing the numerical foundation for solving LCPs with numerical methods that are suitable for computer graphics. This is the void that these course notes tries to fill out – providing a toolbox of solutions for people in physics-based animation. The contribution of these notes is twofold. First, we explain the nature of LCPs and discuss the properties of the LCPs encountered in physics-based animation. Second, we present a range of numerical methods for solving the LCPs. To help make our results available to others we supplement our course notes with Matlab implementations of all iterative methods discussed.

Keywords: Linear Complementarity Problems, Newton Methods, Splitting Methods, Interior Point Methods, Convergence Rate, Performance Study.

Contents

1 Introduction

The linear complementarity problems (LCPs) are notorious in computer graphics as being hard to solve and difficult to understand. However, LCPs are important general purpose models that extend beyond rigid body dynamics where they found initial applications [Bar89, Bar93, Bar94, Bar95]. It is evident that now deformable models, granular materials and fluids are being formulated using LCPs [DDKA06, BBB07, OGRG07, CM11, OTSG09, GZO10, AO11].

Previous work in the field of computer graphics on numerical methods for solving LCPs are sparse. Baraff [Bar94] introduced a Dantzig pivoting method to the graphics community. More recently Lemke's pivoting method has received some attention [Hec04, Kip07, Ebe10]. There are examples of projected Gauss–Seidel (PGS) type methods [Erl07, CA09, GZO10]. These works often overlook the problem that PGS methods may not converge on the problems that are being solved. Recently, a multilevel PGS method was presented [CM11] but did not provide any convergence guarantees. In [OGRG07] a multilevel method for solving the elastic deformation of a deformable model was combined with a PGS-style solver. However, the paper did not address multigrid PGS. Many details on LCPs can be found in applied mathematics textbooks [Mur88, CPS92].

We speculate that LCPs will become even more attractive models on a wider scope in computer graphics once fast and efficient numerical solutions are easily accesible by researchers in simple to implement numerical methods. This is why we wrote these notes.

A supplementary code repository may be found in [Erl11] containing Matlab implementations of all the iterative methods covered in these notes and a few implementations in Python. C++ (uBLAS) and CUSP versions of few selected methods are available by email request.

These course notes assume that the reader is familiar with linear algebra and differential calculus. All numerical methods presented are described in a pseudo-code style independent of any specific programming language, and should be understandable by any computer scientist with rudimentary programming skills.

2 Understanding The LCP

A one-dimensional complementarity problem (CP) can be stated as having two real variables $x, y \in \mathbb{R}$ where we seek to make sure that they always satisfy the complementarity constraint,

$$y > 0 \quad \Rightarrow \quad x = 0 \quad \text{or} \quad x > 0 \quad \Rightarrow y = 0. \tag{1}$$

This results in a flip-flop problem that either one variable is positive and the other is zero or vice versa. This is written compactly in the notation

$$0 \leq y \quad \perp \quad x \geq 0. \tag{2}$$

The solution space forms a corner shape given by the positive x and y axes. If we now extend the problem to include a linear relation between the y and x variables, $y = ax + b$ where $a, b \in \mathbb{R}$ then we have a LCP,

$$y = ax + b, \tag{3a}$$
$$y \geq 0, x \geq 0, \text{ and } xy = 0. \tag{3b}$$

We can conceptually visualize the whole solution space by adding the linear relation on top of the corner shape. Now the solutions for the model are the intersection points between the line and the corner shape. This geometric tool provides us with an approach to study the nature of LCPs. We ask the reader to consider what would happen if $b < 0$ and $a < 0$ or $b = 0$ and $a = 0$? Apparently, the model parameters a and b determine whether we can expect to find a solution and whether a solution is unique or not. If we eliminate the y-variable then the LCP can be written compactly as,

$$ax + b \geq 0, \tag{4a}$$
$$x \geq 0, \tag{4b}$$
$$x\,(ax + b) = 0. \tag{4c}$$

This new form suggests a different approach to finding the solutions. We observe that the complementarity condition (4c) has the familiar form of a quadratic function. Because of the inequalities (4a)- (4b) (called "unilateral constraints"), any feasible x-value will result in a nonnegative value of the quadratic function. The quadratic function will be zero only for a solution of the LCP. In other words, we may rephrase the LCP as solving the optimization problem

$$x^* = \arg\min_x \; x\,(ax + b) \tag{5}$$

subject to

$$x \geq 0 \text{ and } ax + b \geq 0. \tag{6}$$

Following this line of inspiration and exploring the connection to optimization problems, we find the first-order optimality conditions (also known as the Karush-Kuhn-Tucker conditions or KKT conditions for short) for the optimization problem [NW99],

$$x^* = \arg\min_{x \geq 0} \; x\left(\frac{1}{2}ax + b\right), \tag{7}$$

to be (3), where y will be the Lagrange multiplier.

We will show this in detail. Let the objective function be written as $f(x) = x\left(\frac{1}{2}ax + b\right) = \frac{1}{2}ax^2 + bx$ then the Lagrangian is formally defined as

$$\mathcal{L}(x, y) = f(x) - yx, \tag{8}$$

where we let y denote the Lagrange multiplier corresponding to the unilateral constraint $x \geq 0$. The first-order optimality conditions are

$$\nabla_x \mathcal{L}(x, y) = 0, \tag{9a}$$
$$y \geq 0, \tag{9b}$$
$$x \geq 0, \tag{9c}$$
$$yx = 0. \tag{9d}$$

Noting that $\nabla_x \mathcal{L}(x, y) = ax + b - y = 0$ we have $y = ax + b$. Substitution yields

$$ax + b \geq 0 \tag{10}$$
$$x \geq 0 \tag{11}$$
$$x\,(ax + b) = 0 \tag{12}$$

In other words we arrive at the LCP again. This is an important observation. The LCP is not the same as an optimization problem. However, all first-order optimality solutions for the optimization problem above will be a solution for the LCP and vice versa. Again the optimization reformulation gives us insight into whether one can expect to find solutions. For this 1D case we note that whenever $a > 0$ one has a strict convex optimization problem subject to linear constraints. Thus, constraint qualifications are fulfilled and one is guaranteed a solution exists [NW99] (Constraint qualifications are sufficient conditions for an optimization problem such that the tangent cone and the set of linearized feasible directions are the same set. This are necessary regularity conditions that ensure the first-order conditions are well-posed). If $a < 0$ then the objective is unbounded from below and we may get into trouble. Observe the 1D LCP is a combinatorial problem. As soon as one has chosen whether y or x is positive then the problem is reduced to that of a linear relation from which the solution is trivially computed.

Before moving into higher-dimensional spaces we will introduce more ideas on reformulating the LCP into other types of problems. The first reformulation we will present is known as the minimum map reformulation[Pan90, CPS92, Mur88]. Essentially it can be written as

$$h(x, y) = \min(x, y). \tag{13}$$

Here the minimum map function, $\min(x, y)$, is defined as

$$\min(x, y) = \begin{cases} x & \text{if } x < y \\ y & \text{otherwise} \end{cases} \tag{14}$$

Now a solution x^*, y^* for the LCP fulfills $h(x^*, y^*) = 0$. Thus, we have

$$h(x^*, y^*) = 0 \quad \text{iff} \quad 0 \leq y^* \quad \perp \quad x^* \geq 0. \tag{15}$$

This can be proven by a case-by-case analysis as shown here

$h(x,y)$	$y<0$	$y=0$	$y>0$
$x<0$	<0	<0	<0
$x=0$	<0	$=0$	$=0$
$x>0$	<0	$=0$	>0

For the LCP we know that y is a function of x and thus h is essentially only a function of x, we write

$$h(x) = \min(x, ax + b). \tag{16}$$

The benefit of this reformulation of the LCP is that we have converted our problem into a root finding problem. Thus, any solution x^* for $h(x) = 0$ will be a solution for our LCP. Another popular reformulation is based on the Fischer-Burmeister function which is defined as [Fis92, CPS92, Mur88]

$$\phi(x,y) \equiv \sqrt{x^2 + y^2} - x - y. \tag{17}$$

Again we have a similar property as for the minimum map reformulation, namely

$$\phi(x^*, y^*) = 0 \quad \text{iff} \quad 0 \leq y^* \quad \perp \quad x^* \geq 0. \tag{18}$$

As before this can be proven by a case-by-case analysis.

$\phi(x,y)$	$y<0$	$y=0$	$y>0$
$x<0$	>0	>0	>0
$x=0$	>0	$=0$	$=0$
$x>0$	>0	$=0$	<0

Again we have the option of solving our problem by finding the roots of $\phi(x) = 0$. One may believe that one can simply plug the $h(x)$ or $\phi(x)$ into the Newton–Raphson method and find a solution. The problem with h and ϕ is that they are nonsmooth functions implying that for certain points one can not compute the derivatives $\frac{\partial h(x)}{\partial x}$ and $\frac{\partial \phi(x)}{\partial x}$. To circumvent the problem we will need nonsmooth analysis that allows us to compute a generalized Jacobian which we can use to create a generalized Newton method.

3 Going to Higher Dimensions

Having gained familiarity with the one-dimensional LCP we will now extend the ideas to higher dimensions. Let $\mathbf{b}, \mathbf{x} \in \mathbb{R}^n$ and $\mathbf{A} \in \mathbb{R}^{n \times n}$ so $\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{b}$. For the n-dimensional LCP we have for all $i \in [1..n]$,

$$\mathbf{x}_i \geq 0, \tag{19a}$$
$$(\mathbf{A}\mathbf{x} + \mathbf{b})_i \geq 0, \tag{19b}$$
$$\mathbf{x}_i(\mathbf{A}\mathbf{x} + \mathbf{b})_i = 0. \tag{19c}$$

We can write this compactly in matrix-vector notation as

$$\mathbf{x} \geq 0, \tag{20a}$$
$$(\mathbf{A}\mathbf{x} + \mathbf{b}) \geq 0, \tag{20b}$$
$$\mathbf{x}^T(\mathbf{A}\mathbf{x} + \mathbf{b}) = 0. \tag{20c}$$

Observe that $\geq$ implies that the inequality holds element-wise. That is $\mathbf{x} \geq \mathbf{0}$ implies $\mathbf{x}_i \geq 0$ for all i. We may now rediscover the reformulations we have introduced earlier for the one-dimensional case. Assuming a symmetric $\mathbf{A}$-matrix the Quadratic Programming (QP) reformulation becomes

$$\mathbf{x}^* = \arg\min_{\mathbf{x} \geq \mathbf{0}} f(\mathbf{x}) \tag{21}$$

where $f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{b}^T\mathbf{x}$. The first order optimality conditions of the optimization problem is the LCP [NW99],

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{b}, \tag{22a}$$
$$\mathbf{y} \geq \mathbf{0}, \tag{22b}$$
$$\mathbf{x} \geq \mathbf{0}, \tag{22c}$$
$$\mathbf{x}^T\mathbf{y} = 0. \tag{22d}$$

Applying the minimum map reformulation in an element-wise manner results in the nonsmooth root search problem,

$$\mathbf{H}(\mathbf{x}) = \mathbf{H}(\mathbf{x}, \mathbf{y}) = \begin{bmatrix} h(\mathbf{x}_1, \mathbf{y}_1) \\ \dots \\ h(\mathbf{x}_n, \mathbf{y}_n) \end{bmatrix} = \mathbf{0}. \tag{23}$$

The Fischer-Burmeister function can be applied individually to each complementarity constraint to create the root search problem,

$$\mathbf{F}(\mathbf{x}) = \mathbf{F}(\mathbf{x}, \mathbf{y}) = \begin{bmatrix} \phi(\mathbf{x}_1, \mathbf{y}_1) \\ \vdots \\ \phi(\mathbf{x}_n, \mathbf{y}_n) \end{bmatrix} = \mathbf{0}. \tag{24}$$

From a practical viewpoint we are now concerned with how we can numerically solve these various problem formulations which will be the topic of Section 5. Before we study the numerical methods we will briefly in Section 4 motivate linear complementarity problem cases found in physics-based animation through examples.

4 Examples of LCPs in Physics-based Animation

We will in the following two subsections show two examples of LCP models one for contact forces and another for fluid-solid wall boundaries.

4.1 The Contact Force Model Example

We will present a LCP model for frictional contact force computations [Löt84, ST96, AP97]. It should be noted that contact forces can be modeled in other ways than using LCPs [BDCDA11, DBDB11, AT08]. We refer the interested

reader to [BETC12] for complete references on alternatives. However, a lot of recent work in the field of graphics is using LCPs [KSJP08, CA09, AFC+10, SKV+12, TBV12]. Here we do not seek to cover all alternatives but settle for motivating how a simulation problem can be written as a LCP problem. Rigorous modeling details may be found in [BETC12].

To simplify notation we will without loss of generality present the model abstractly as though we were considering a single contact point. To model non-penetration one use the complementarity condition

$$0 \leq \mathbf{v}_n \quad \perp \quad \lambda_n \geq 0 \tag{25a}$$

where $\mathbf{v}_n$ is the normal component of the relative contact velocity and λ_n is the magnitude of the normal contact impulse. This means if there is a separation $\mathbf{v}_n > 0$ then there can be no normal contact impulse. On the other hand if there is a normal contact impulse $\lambda_n > 0$ this is non-sticking and there must be a resting contact $\mathbf{v}_n = 0$.

Stewart and Trinkle, Anitescu and Potra [ST96, AP97] among others linearized the 3D friction model using a polyhedral cone. The polyhedral cone is given by a positive span of K unit vectors $\mathbf{t}_i$. Let λ_n be the magnitude of the normal impulse and $\lambda_t = \begin{bmatrix}\lambda_{t_1} & \cdots & \lambda_{t_K}\end{bmatrix}^T$ the vector of friction impulses. The linearized model is

$$0 \leq \mathbf{v}_n \quad \perp \quad \lambda_n \geq 0, \tag{26a}$$

$$0 \leq \beta\mathbf{e} + \mathbf{v}_t \quad \perp \quad \lambda_t \geq 0, \tag{26b}$$

$$0 \leq \left(\mu\lambda_n - \sum_i \lambda_{t_i}\right) \quad \perp \quad \beta \geq 0, \tag{26c}$$

where $\mathbf{e}$ is a K dimensional vector of ones. The first complementarity constraint models the non-penetration constraint as before. The second equation makes sure that in case we do have friction $\lambda_{t_i} > 0$ for some i then β will estimate the maximum sliding velocity along the $\mathbf{t}_i$'s directions. Observe this equation is a K-dimensional vector equation. Its main purpose is to choose the $\mathbf{t}_i$ direction that best approximates the direction of maximum dissipation. The last equation makes sure the friction force is bounded by the Coulomb friction cone. Notice that if $\beta > 0$ the last equation will force the friction force to lie on the boundary of the polyhedral friction cone. If $\beta = 0$ the two last equations model static friction. That is no sliding can occur and any friction force inside the friction cone is feasible.

Due to the positive span of $\mathbf{t}_i$ one usually have several $\mathbf{v}_{t_i} \neq 0$ for sliding motion. However, the model will pick only one λ_{t_i} to be non-zero. The t_i-direction chosen by the model is the one mostly opposing the sliding direction. Only in the rare case where the sliding direction is symmetrically between t_i-directions the model may pick two positive λ_{t_i} values.

Observe that $\sum_i \lambda_{t_i} = \mathbf{e}^T\lambda_t$ and we have $\mathbf{v} = \begin{bmatrix}\mathbf{v}_n & \mathbf{v}_t\end{bmatrix}^T$. From the discretization of the Newton–Euler equations we have the contact velocity-impulse

relation $\mathbf{v} = \mathbf{B}\lambda + \mathbf{b}$. The term $\mathbf{b}$ contains initial velocity terms hence $\mathbf{v}$ is the final velocity obtained by applying the impulse λ. Using all this we can write the final matrix form of the contact model as,

$$\mathbf{0} \leq \underbrace{\begin{bmatrix} \mathbf{B}_{nn} & \mathbf{B}_{nt} & 0 \\ \mathbf{B}_{tn} & \mathbf{B}_{tt} & \mathbf{e} \\ \mu & -\mathbf{e}^T & 0 \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} \lambda_n \\ \lambda_t \\ \beta \end{bmatrix}}_{\mathbf{x}} + \underbrace{\begin{bmatrix} \mathbf{b}_n \\ \mathbf{b}_t \\ 0 \end{bmatrix}}_{\mathbf{b}} \quad \perp \quad \underbrace{\begin{bmatrix} \lambda_n \\ \lambda_t \\ \beta \end{bmatrix}}_{\mathbf{x}} \geq \mathbf{0} \tag{27}$$

where $\mathbf{B}_{nn} = \mathbf{J}_n\mathbf{M}^{-1}\mathbf{J}_n^T$, $\mathbf{B}_{nt} = \mathbf{B}_{tn}^T = \mathbf{J}_n\mathbf{M}^{-1}\mathbf{J}_t^T$, $\mathbf{B}_{tt} = \mathbf{J}_t\mathbf{M}^{-1}\mathbf{J}_t^T$, $\mathbf{b}_n = \mathbf{J}_n\mathbf{M}^{-1}\mathbf{F}$, and $\mathbf{b}_t = \mathbf{J}_t\mathbf{M}^{-1}\mathbf{F}$. Here $\mathbf{M}$ is the generalized mass matrix, $\mathbf{J}_n$ and $\mathbf{J}_t$ are the normal and tangential parts of the contact Jacobian such that $\mathbf{f}_n = -\mathbf{J}_n^T\lambda_n$ and $\mathbf{f}_t = -\mathbf{J}_t^T\lambda_t$, and $\mathbf{F}$ is a vector including external loads and gyroscopic forces. Observe that $\mathbf{A}$ is non-symmetric and has a zero-diagonal block. Further the subblock $\mathbf{B}$ is symmetric and positive-semi-definite (PSD) matrix. As will be clear from Section 5 this implies that we can not use PGS for the contact LCP model. Rather we can use either Lemke's method or a Newton-based method such as the one in Section 5.5.

There exist an alternative complementarity problem formulation which drops the β-part of the model by ignoring principle of maximum dissipation. The resulting model is no longer a LCP, but one could apply a splitting based PGS method for this alternative model [Erl07]. This is similar to the method we present in Section 5.2. The alternative model is physically flawed in the sense that friction directions are decoupled and no convergence guarantees can be given for the PGS-type method [SNE09, SNE10].

4.2 The Fluid LCP Model Example

Recently, LCPs are being used to model fluid-solid wall boundary conditions [BBB07, CM11]. This is a recent approach and the literature is still sparse on examples. In physics-based animation most works use the incompressible Euler equations [Bri08]

$$\rho\frac{\partial \mathbf{u}}{\partial t} = -\left(\mathbf{u}\cdot\nabla\right)\mathbf{u} - \nabla p - \mathbf{f}, \tag{28a}$$

$$\nabla\cdot\mathbf{u} = 0, \tag{28b}$$

where ρ is mass density, $\mathbf{u}$ is the velocity field, p is the pressure field and $\mathbf{f}$ is the external force density. Traditionally one applies the boundary conditions $p = 0$ on free surfaces between fluid and vacuum and $\mathbf{u}\cdot\mathbf{n} = 0$ between the fluid and a static solid wall with unit outward normal $\mathbf{n}$. For simplicity we here just present ideas for a single phase flow in vacuum. The ideas trivially generalize to multiphase flow and dynamic solid wall boundary conditions [BBB07, CM11]. In physics-based animation coarse grids are used to keep the computational cost down. This causes a problem with the traditional solid wall boundary condition $\mathbf{u}\cdot\mathbf{n} = 0$. Namely that cell-size thick layers of fluid are getting stuck on walls.

This appears visually unrealistic. Thus, it has been proposed to change the solid wall boundary condition to,

$$0 \leq p \quad \perp \quad \mathbf{u} \cdot \mathbf{n} \geq 0. \tag{29}$$

This allows the fluid to separate from the wall. The condition $\mathbf{u} \cdot \mathbf{n} > 0$ enforces $p = 0$ making the interface act like a free surface. On the other hand if $\mathbf{u} \cdot \mathbf{n} = 0$ then the fluid is at rest at the wall and there must be a pressure $p > 0$ acting on the fluid to keep it at rest.

The current trend is to spatially discretize the equations of motion on a staggered regular grid using finite difference approximations of the spatial derivatives. For the temporal derivative one deals with the partial differential equation using a fractional step method (known as operator splitting) [Sta99]. This means that in the last sub-step of the fractional step method one is solving,

$$\mathbf{u}^{n+1} = \mathbf{u}' - \frac{\Delta t}{\rho} \nabla p, \tag{30a}$$

$$\nabla \cdot \mathbf{u}^{n+1} = 0, \tag{30b}$$

where $\mathbf{u}^{n+1}$ is the final divergence free velocity of the fluid and $\mathbf{u}'$ is the fluid velocity obtained from the previous step in the fractional step method. The time-step is given by Δt. Substituting the first equation into the second yields

$$\nabla \cdot \mathbf{u}^{n+1} = \nabla \cdot \mathbf{u}' - \frac{\Delta t}{\rho} \nabla^2 p = 0. \tag{31}$$

Introducing the spatial discretization, we obtain the Poisson equation which we for notational convenience write as

$$\mathbf{A}\mathbf{p} + \mathbf{b} = \mathbf{0}, \tag{32}$$

where $\mathbf{p}$ is the vector of all cell-centered pressure values and $\mathbf{A} \equiv \left\{ -\frac{\Delta t}{\rho} \nabla^2 \right\}$ and $\mathbf{b} \equiv \{\nabla \cdot \mathbf{u}'\}$. The matrix $\mathbf{A}$ is a symmetric diagonal banded matrix. In 2D it will have 5 bands when using a 5-point stencil, in 3D it will have 7 bands for a 7-point stencil. For regular grids all off diagonal bands have the same value. Further, $\mathbf{A}$ is known to be a PSD matrix, but adding the boundary condition $\mathbf{p} = \mathbf{0}$ ensures that a unique solution can be found. Once the pressure vector has been computed, it can be used to compute the last step of the fractional step method (30a).

Let us revisit the complementarity problem arising from the modified boundary condition and examine what happens if $\mathbf{u}^{n+1} \cdot \mathbf{n} > 0$ at a solid wall boundary. To start the analysis we will examine what happens with (28b) in an arbitrary small control volume V around a solid wall boundary point,

$$\int_V \nabla \cdot \mathbf{u}^{n+1} \, dV = \oint_S \mathbf{u}^{n+1} \cdot \mathbf{n} \, dS > 0. \tag{33}$$

The last inequality follows from the assumption that $\mathbf{u}^{n+1} \cdot \mathbf{n} > 0$. This means that if we pick the row of the discrete Poisson equation that corresponds to the solid wall boundary point, we obtain (for the j^{th} row)

$$\mathbf{A}_{j*}\mathbf{p} + \mathbf{b}_j > 0. \tag{34}$$

If on the other hand $\mathbf{u}^{n+1} \cdot \mathbf{n} = 0$ at the solid wall, then we rediscover $\mathbf{A}_{j*}\mathbf{p} + \mathbf{b}_j = 0$. Although we skipped all the details of the discretization it should be intuitively clear that the pressure solve for the new modified boundary conditions is given by the LCP,

$$0 \leq \mathbf{p} \quad \perp \quad \mathbf{A}\mathbf{p} + \mathbf{b} \geq 0. \tag{35}$$

Having motivated LCP models through two examples we may now turn our attention towards developing numerical methods for solving such LCP models.

5 The Numerical Methods

We may now embark on the real issue at hand – how to make robust, efficient and fast methods for solving the kind of LCPs we encounter in physics-based animation.

5.1 Pivoting Methods

We will exploit the combinatorial nature of the LCP to outline a guessing approach for finding a solution. In principle if all possible guesses are tested then one obtains a naive direct enumeration method that will find all solutions. By algebraic manipulation on $\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{b}$,

$$\begin{bmatrix} \mathbf{I} & -\mathbf{A} \end{bmatrix} \begin{bmatrix} \mathbf{y} \\ \mathbf{x} \end{bmatrix} = \mathbf{b}. \tag{36}$$

We define the integer index set $\mathcal{I} \equiv \{1, \ldots, n\}$ of all indices. Next we use two index sets one of free variables $\mathbf{y}_i > 0$ and one of active variables $\mathbf{y}_i = 0$,

$$\mathcal{F} \equiv \{i \mid \mathbf{y}_i > 0\} \text{ and } \mathcal{A} \equiv \{i \mid \mathbf{x}_i > 0\}. \tag{37}$$

Without loss of generality and to keep our presentation simple we assume that strict complementarity holds which means we never simultaneously have $\mathbf{y}_i = 0$ and $\mathbf{x}_i = 0$. This means we can assume that $\mathcal{F} \cap \mathcal{A} = \emptyset$ and $\mathcal{F} \cup \mathcal{A} = \mathcal{I}$. The idea is to create a method that can verify if a guess of $\mathcal{F}$ and $\mathcal{A}$ is a solution for the given LCP. Now using the index sets we make the partitioning,

$$\underbrace{\begin{bmatrix} \mathbf{I}_{*\mathcal{F}} & -\mathbf{A}_{*\mathcal{A}} \end{bmatrix}}_{\mathbf{C}} \underbrace{\begin{bmatrix} \mathbf{y}_{\mathcal{F}} \\ \mathbf{x}_{\mathcal{A}} \end{bmatrix}}_{\mathbf{s}} = \mathbf{b}. \tag{38}$$

where $\mathbf{I}_{*\mathcal{F}}$ and $\mathbf{A}_{*\mathcal{A}}$ are the sub matrices given by the column indices $\mathcal{F}$ and $\mathcal{A}$. Our problem is now simplified to verify if the linear programming (LP) problem

$$\mathbf{Cs} = \mathbf{b} \quad \text{subject to} \quad \mathbf{s} \geq \mathbf{0} \tag{39}$$

has a solution[1]. This is the same as testing if $\mathbf{b}$ is in the positive cone of $\mathbf{C}$. That is,

$$\mathbf{b} \in \{\mathbf{Cs} \mid \mathbf{s} \geq \mathbf{0}\}. \tag{40}$$

Traditionally $\mathbf{C}$ is called a complementarity matrix. Observe that it is constructed by picking columns from either $\mathbf{A}$ or $\mathbf{I}$. Clearly we can make 2^n different complementarity matrices. For low-dimensional LCPs this suggests a geometric approach to find a solution for a LCP.

In the worst case the time complexity of guessing would be $\mathcal{O}(n^3 2^n)$ which is not computationally efficient. Another strategy is to be clever in making new guesses, for example, by applying a pivoting strategy that moves an index from one set to the other or some strategy that builds up the index sets incrementally. Among direct methods for LCPs based on pivoting are the Lemke method (a Matlab implementation can be found from CPNET) and the Keller method [CPS92, Lac03].

The pivoting methods are capable of finding an accurate solution to the LCP whereas the iterative methods we cover in Section 5.2- 5.5 only find approximate solutions. The accuracy is at the expense of having to form the $\mathbf{A}$-matrix in the first place whereas the iterative methods often can exploit a factorization of the $\mathbf{A}$-matrix. Even if $\mathbf{A}$ is sparse the fill in of the inverse matrix $\mathbf{A}_{\mathcal{AA}}^{-1}$ may be dense implying that the worst case storage complexity of pivoting methods is of $\mathcal{O}(n^2)$ complexity.

The pivoting methods have some similarity with active set methods for constrained QP problems [NW99]. In the case of symmetric PSD $\mathbf{A}$-matrices, one may consider restating the problem as a QP problem rather than implementing the pivoting method. This is especially beneficial due to the availability of QP solvers such as MOSEK, CPLEX, LANCELOT, SQP, SNOPT and many more.

5.1.1 Incremental Pivoting "Baraff Style"

In the field of computer graphics Baraff presented an incremental pivoting method [Bar94]. This method incrementally builds up the index sets while keeping the complementarity constraints as invariants. In each iteration the method computes $\mathbf{A}_{\mathcal{AA}}^{-1}$. Whenever $\mathbf{A}$ is a symmetric positive definite (PD) $\mathbf{A}_{\mathcal{AA}}^{-1}$ exist. Baraff reported that even when $\mathbf{A}$ is PSD (which is often the case in practice due to redundant contact constraints), he was able to compute $\mathbf{A}_{\mathcal{AA}}^{-1}$. Baraff proves that the inner pivoting loop of his method only can be performed a finite number of times as the index set $\mathcal{A}$ is never repeated. Thus, the cost of the inner loop is at worst that of computing $\mathbf{A}_{\mathcal{AA}}^{-1}$ which is $\mathcal{O}(n^3)$. The outer loop of

[1] We could more precisely have written $\mathbf{s} > \mathbf{0}$ but $\mathbf{s} \geq \mathbf{0}$ also covers the more general non-strict case.

this method runs for at most n iterations yielding a pessimistic time complexity of $\mathcal{O}(n^4)$. Noticing that the pivot step only swaps one index and therefore only changes the size of $\mathcal{A}$ by one it is clear that an incremental factorization method can be used for computing $\mathbf{A}_{\mathcal{A},\mathcal{A}}^{-1}$. There exist incremental factorization running in $\mathcal{O}(n^2)$ time complexity. Thus, a more realistic overall time complexity for the pivoting method is $\mathcal{O}(n^3)$.

For completeness we outline the pivoting method here. In the k^{th} iteration a new index will be selected from the current set of unprocessed indices, $\mathcal{U} \equiv \mathcal{I} \setminus \{\mathcal{F} \cup \mathcal{A}\}$. The index sets $\mathcal{F}$ and $\mathcal{A}$ are initially both empty. Throughout, the complementarity conditions are kept as invariants. We will use superscript k to denote the values at a given iteration number. For any unprocessed index $j \in \mathcal{U}$ we implicitly assume $\mathbf{x}_j^k = 0$. Initially in the k^{th} iteration we use the partitioning

$$\begin{bmatrix} \mathbf{y}_{\mathcal{A}}^k \\ \mathbf{y}_{\mathcal{F}}^k \\ \mathbf{y}_{\mathcal{U}}^k \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{\mathcal{A},\mathcal{A}} & \mathbf{A}_{\mathcal{A},\mathcal{F}} & \mathbf{A}_{\mathcal{A},j} \\ \mathbf{A}_{\mathcal{F},\mathcal{A}} & \mathbf{A}_{\mathcal{F},\mathcal{F}} & \mathbf{A}_{\mathcal{F},j} \\ \mathbf{A}_{\mathcal{U},\mathcal{A}} & \mathbf{A}_{\mathcal{U},\mathcal{F}} & \mathbf{A}_{\mathcal{U},\mathcal{U}} \end{bmatrix} \begin{bmatrix} \mathbf{x}_{\mathcal{A}}^k \\ \mathbf{x}_{\mathcal{F}}^k \\ \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{b}_{\mathcal{A}} \\ \mathbf{b}_{\mathcal{F}} \\ \mathbf{b}_{\mathcal{U}} \end{bmatrix}.$$

The next candidate index to be processed in the method is selected as the index $j \in \mathcal{U}$ that minimize $\mathbf{y}_j^k$ as this corresponds to an index in $\mathcal{U}$ with a most violated complementarity constraint. If for the minimum value the condition $\mathbf{y}_j^k \geq 0$ is fulfilled, the method terminates as this would indicate that all the remaining unprocessed indices trivially fulfill the complementarity conditions. If no unique feasible minimum exists, one may pick a minimizing index at random. In the k^{th} iteration, we use the partitioning and keep the complementarity conditions as invariants implying $\mathbf{y}_{\mathcal{A}}^{k+1} = \mathbf{0}$ and $\mathbf{x}_{\mathcal{F}}^{k+1} = \mathbf{0}$, so

$$\begin{bmatrix} \mathbf{0} \\ \mathbf{y}_{\mathcal{F}}^{k+1} \\ \mathbf{y}_j^{k+1} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{\mathcal{A},\mathcal{A}} & \mathbf{A}_{\mathcal{A},\mathcal{F}} & \mathbf{A}_{\mathcal{A},j} \\ \mathbf{A}_{\mathcal{F},\mathcal{A}} & \mathbf{A}_{\mathcal{F},\mathcal{F}} & \mathbf{A}_{\mathcal{F},j} \\ \mathbf{A}_{j,\mathcal{A}} & \mathbf{A}_{j,\mathcal{F}} & \mathbf{A}_{j,j} \end{bmatrix} \begin{bmatrix} \mathbf{x}_{\mathcal{A}}^{k+1} \\ \mathbf{0} \\ \mathbf{x}_j^{k+1} \end{bmatrix} + \begin{bmatrix} \mathbf{b}_{\mathcal{A}} \\ \mathbf{b}_{\mathcal{F}} \\ \mathbf{b}_j \end{bmatrix}.$$

The changes in $\mathbf{y}_{\mathcal{F}}$ and $\mathbf{x}_{\mathcal{A}}$ with respect to $\mathbf{x}_j^{k+1} > 0$ are given by

$$\begin{aligned} \mathbf{x}_{\mathcal{A}}^{k+1} &= \mathbf{x}_{\mathcal{A}}^k + \Delta\mathbf{x}_{\mathcal{A}}\mathbf{x}_j^{k+1}, \\ \mathbf{y}_{\mathcal{F}}^{k+1} &= \mathbf{y}_{\mathcal{F}}^k + \Delta\mathbf{y}_{\mathcal{F}}\mathbf{x}_j^{k+1}, \\ \mathbf{y}_j^{k+1} &= \mathbf{y}_j^k + \Delta\mathbf{y}_j\mathbf{x}_j^{k+1} \end{aligned}$$

where

$$\begin{aligned} \Delta\mathbf{x}_{\mathcal{A}} &= -\mathbf{A}_{\mathcal{A},\mathcal{A}}^{-1}\mathbf{A}_{\mathcal{A},j}, \\ \Delta\mathbf{x}_{\mathcal{A}} &= -\mathbf{A}_{\mathcal{A},\mathcal{A}}^{-1}\mathbf{A}_{\mathcal{A},j}, \\ \Delta\mathbf{y}_j &= -\mathbf{A}_{j,j} - \mathbf{A}_{j,\mathcal{A}}\mathbf{A}_{\mathcal{A},\mathcal{A}}^{-1}\mathbf{A}_{\mathcal{A},j}. \end{aligned}$$

The idea is to increase $\mathbf{x}_j^{k+1}$ as mush as possible without breaking any of the complementarity constraints. Thus, $\mathbf{x}_j^{k+1}$ is limited by the blocking constraint

set

$$\mathcal{B}_{\mathcal{A}} \equiv \left\{ \frac{-\mathbf{x}_q^k}{\Delta \mathbf{x}_q} \;\middle|\; q \in \mathcal{A} \;\wedge\; \Delta \mathbf{x}_q < 0 \right\}, \tag{42a}$$

$$\mathcal{B}_{\mathcal{F}} \equiv \left\{ \frac{-\mathbf{y}_r^k}{\Delta \mathbf{y}_r} \;\middle|\; r \in \mathcal{F} \;\wedge\; \Delta \mathbf{y}_r < 0 \right\}. \tag{42b}$$

If no blocking constraints exist, $\mathbf{x}_j^{k+1}$ is unbounded by $\mathcal{A}$ and $\mathcal{F}$. Thus, each partition results in the bounds

$$\mathbf{x}_j^{\mathcal{A}} = \begin{cases} \infty & : \mathcal{B}_{\mathcal{A}} = \emptyset \\ \min \mathcal{B}_{\mathcal{A}} & ; \end{cases}, \tag{43a}$$

$$\mathbf{x}_j^{\mathcal{F}} = \begin{cases} \infty & : \mathcal{B}_{\mathcal{F}} = \emptyset \\ \min \mathcal{B}_{\mathcal{F}} & ; \end{cases}, \tag{43b}$$

$$\mathbf{x}_j^{j} = \begin{cases} \frac{-\mathbf{y}_j^k}{\Delta \mathbf{y}_j} & ; \Delta \mathbf{y}_j < 0 \\ 0 & ; \end{cases}. \tag{43c}$$

The solution for the value of $\mathbf{x}_j^{k+1}$ will be the minimum bound. If a blocking constraint is found from $\mathcal{B}_{\mathcal{A}}$, a pivot operation is initiated moving the blocking index from $\mathcal{A}$ to $\mathcal{F}$ and vice versa if a blocking constraint is found in $\mathcal{B}_{\mathcal{F}}$.

The blocking constraint sets are changed as the active and free index sets $\mathcal{A}$ and $\mathcal{F}$ are changed by a pivoting operation. This implies that one could increase $\mathbf{x}_j^{k+1}$ further after a pivoting step. Thus, we will continue to look for blocking constraints and perform pivoting on them until no more blocking constraints exist. Depending on the final value of $\mathbf{x}_j^{k+1}$, index j is assigned to either $\mathcal{F}$ or $\mathcal{A}$.

5.2 Splitting Methods

We introduce the splitting $\mathbf{A} = \mathbf{M} - \mathbf{N}$. We let $\mathbf{c}^k = \mathbf{b} - \mathbf{N}\mathbf{x}^k$, and LCP (20) becomes

$$\mathbf{M}\mathbf{x}^{k+1} + \mathbf{c}^k \geq \mathbf{0}, \tag{44a}$$

$$\mathbf{x}^{k+1} \geq \mathbf{0}, \tag{44b}$$

$$(\mathbf{x}^{k+1})^T (\mathbf{M}\mathbf{x}^{k+1} + \mathbf{c}^k) = 0. \tag{44c}$$

This results in a fixed-point formulation where we hope that for a suitable choice of $\mathbf{M}$ and $\mathbf{N}$ the complementarity subproblem might be easier to solve than the original problem. Imagine for instance picking $\mathbf{M}$ as the diagonal of $\mathbf{A}$. This choice decouples all variables and we have a problem of n independent 1D LCPs. Later we list other known choices for $\mathbf{M}$. The splitting method can be summarized as

Step 0 Initialization, set $k = 0$ and choose an arbitrary nonnegative $\mathbf{x}^0 \geq \mathbf{0}$.

Step 1 Given $\mathbf{x}^k \geq \mathbf{0}$ solve the LCP (44).

Step 2 If $\mathbf{x}^{k+1}$ satisfy some stopping criteria then stop, otherwise set $k \leftarrow k+1$ and go to step 1.

The splitting is often chosen such that $\mathbf{M}$ is a Q-matrix [2]. This means that $\mathbf{M}$ belongs to the matrix class of matrices where the corresponding LCP has a solution for all vectors $\mathbf{c}^k$. Clearly if $\mathbf{x}^{k+1}$ is a solution for (44) and we have $\mathbf{x}^{k+1} = \mathbf{x}^k$ then by substitution into the subproblem given by (44) we see that $\mathbf{x}^{k+1}$ is a solution of the original problem (20).

Next we will use the minimum map reformulation on the complementarity subproblem that is equivalent to

$$\min(\mathbf{x}^{k+1}, \mathbf{M}\mathbf{x}^{k+1} + \mathbf{c}^k) = \mathbf{0}. \tag{45}$$

Subtract $\mathbf{x}^{k+1}$ and multiply by minus one,

$$\max(\mathbf{0}, -\mathbf{M}\mathbf{x}^{k+1} - \mathbf{c}^k + \mathbf{x}^{k+1}) = \mathbf{x}^{k+1}. \tag{46}$$

Again we re-discover a fixed-point formulation. Let us perform a case-by-case analysis of the i^{th} component. If we play the mind game and assume

$$\left(\mathbf{x}^{k+1} - \mathbf{M}\mathbf{x}^{k+1} - \mathbf{c}^k\right)_i < 0 \tag{47}$$

then we must have $\mathbf{x}_i^{k+1} = 0$. Otherwise our assumption is false and we must have

$$\left(\mathbf{x}^{k+1} - \mathbf{M}\mathbf{x}^{k+1} - \mathbf{c}^k\right)_i = \mathbf{x}_i^{k+1}. \tag{48}$$

That is

$$(\mathbf{M}\mathbf{x}^{k+1})_i = \mathbf{c}_i^k. \tag{49}$$

For a suitable choice of $\mathbf{M}$ and back-substitution of $\mathbf{c}^k = \mathbf{b} - \mathbf{N}\mathbf{x}^k$ we have

$$\left(\mathbf{M}^{-1}\left(\mathbf{N}\mathbf{x}^k - \mathbf{b}\right)\right)_i = \mathbf{x}_i^{k+1}. \tag{50}$$

Combining it all we have derived the closed form solution for the complementarity subproblem,

$$\max\left(\mathbf{0}, \left(\mathbf{M}^{-1}\left(\mathbf{N}\mathbf{x}^k - \mathbf{b}\right)\right)\right) = \mathbf{x}^{k+1}. \tag{51}$$

Iterative schemes like these are often termed *projection methods*. The reason for this is that if we introduce the vector $\mathbf{z}^k = \mathbf{M}^{-1}\left(\mathbf{N}\mathbf{x}^k - \mathbf{b}\right)$ then

$$\mathbf{x}^{k+1} = \max\left(\mathbf{0}, \mathbf{z}^k\right). \tag{52}$$

That is, the $k+1$ iterate is obtained by projecting the vector $\mathbf{z}^k$ onto the positive octant. In a practical implementation one would rewrite the matrix equation (52) into a for loop that sweeps over the vector components and updates the $\mathbf{x}$-vector in place.

[2] Q-matrix means the LCP given by $\mathbf{M}$ and $\mathbf{c}$ has a solution for all values of $\mathbf{c}$.

One would want to use a clever splitting such that the inversion of $\mathbf{M}$ is computationally cheap. Letting $\mathbf{L}$, $\mathbf{D}$ and $\mathbf{U}$ be the strict lower, diagonal and strict upper parts of $\mathbf{A}$, then three popular choices are: the projected Jacobi method $\mathbf{M} = \mathbf{D}$ and $\mathbf{N} = \mathbf{L} + \mathbf{U}$, the projected Gauss–Seidel (PGS) method $\mathbf{M} = (\mathbf{L}+\mathbf{D})$ and $\mathbf{N} = \mathbf{U}$, and the projected Successive Over Relaxation (PSOR) method $\mathbf{M} = (\mathbf{D} + \lambda\mathbf{L})$ and $\mathbf{N} = ((1-\lambda)\,\mathbf{D} - \lambda\mathbf{U})$ where $0 \leq \lambda \leq 2$ is the relaxation parameter. More about this parameter in Section 5.3.

It is worthwhile to note that $\mathbf{A}$ must at least have nonzero diagonal for these splittings to work. As far as we know there exist no convergence proofs in the general case of $\mathbf{A}$ being arbitrary. However, given appropriate assumptions on $\mathbf{A}$ such as being a contraction mapping or symmetric, one can make proofs of global convergence [CPS92, Mur88]. In section 5.3 we take a different approach to deriving the same iterative schemes. Here it follows by construction that if $\mathbf{A}$ is symmetric and PSD then the splitting schemes will always converge. For non-symmetric matrices one may experience divergence.

5.3 PGS and PSOR from QPs

In our second approach for deriving the PGS and PSOR iterative methods, we will make use of the QP reformulation. Our derivation follows in the footsteps of [Man84]. The reformulation allows us to prove convergence properties of the PGS and PSOR methods. We assume that $\mathbf{A}$ is symmetric and PSD. Then, the LCP can be restated as a minimization problem of a constrained convex QP problem

$$\mathbf{x}^* = \arg\min_{\mathbf{x}\geq\mathbf{0}} f(\mathbf{x}) \tag{53}$$

where $f(\mathbf{x}) \equiv \frac{1}{2}\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{x}^T\mathbf{b}$. Given the i^{th} unit axis vector $\hat{e}^i$ where $\hat{e}^i_j = 0$ for all $j \neq i$ and $\hat{e}^i_i = 1$ then the i^{th} relaxation step consists in solving the one-dimensional problem

$$\tau^* = \arg\min_{\mathbf{x}\geq\mathbf{0}} f(\mathbf{x} + \tau\hat{e}^i) \tag{54}$$

and then setting $\mathbf{x} \leftarrow \mathbf{x} + \tau\hat{e}^i$. One relaxation cycle consists of one sequential sweep over all i^{th} components.

The one-dimensional objective function can be rewritten as

$$\begin{aligned} f(\mathbf{x} + \tau\hat{e}^i) &= \frac{1}{2}(\mathbf{x} + \tau\hat{e}^i)^T\mathbf{A}(\mathbf{x} + \tau\hat{e}^i) + (\mathbf{x} + \tau\hat{e}^i)^T\mathbf{b}, \\ &= \frac{1}{2}\tau^2\mathbf{A}_{ii} + \tau(\underbrace{\mathbf{A}\mathbf{x} + \mathbf{b}}_{\mathbf{r}})_i + f(\mathbf{x}). \end{aligned}$$

We find the unconstrained minimizer $\tau_u = -\frac{\mathbf{r}_i}{\mathbf{A}_{ii}}$. Considering the constraint $\mathbf{x}_i + \tau \geq 0$ we find the constrained minimizer to be $\tau_c = \max(\tau_u, -\mathbf{x}_i)$ which yields the final update rule for the relaxation step

$$\mathbf{x}_i \leftarrow \max\left(0, \mathbf{x}_i - \frac{\mathbf{r}_i}{\mathbf{A}_{ii}}\right). \tag{56}$$

This is algebraically equivalent to the i^{th} component in the PGS update (52). Consider the polynomial $g(\tau) \equiv \frac{1}{2}\tau^2 \mathbf{A}_{ii} + \tau \mathbf{r}_i$. We know $\mathbf{A}_{ii} > 0$ so the legs of the polynomial are pointing upwards. The polynomial has one trivial root $\tau = 0$ and a minimum at $\tau = -\frac{\mathbf{r}_i}{\mathbf{A}_{ii}}$ where $g\left(-\frac{\mathbf{r}_i}{\mathbf{A}_{ii}}\right) = -\frac{\mathbf{r}_i^2}{\mathbf{A}_{ii}} < 0$. The other root is found at $\tau = -2\frac{\mathbf{r}_i}{\mathbf{A}_{ii}}$. Thus, any τ value in the interval between the two roots has the property

$$\tau_\lambda = -\lambda \frac{\mathbf{r}_i}{\mathbf{A}_{ii}} \Rightarrow g(\tau_\lambda) < 0, \quad \forall \lambda \in [0..2]. \tag{57}$$

It follows that

$$f(\mathbf{x} + \tau_\lambda \hat{e}^i) = g(\tau_\lambda) + f(\mathbf{x}) \leq f(\mathbf{x}), \quad \forall \lambda \in [0..2] \tag{58}$$

with equality if $\tau_\lambda = 0$. This results in the over relaxed version

$$\mathbf{x}_i \leftarrow \max\left(0, \mathbf{x}_i - \lambda \frac{\mathbf{r}_i}{\mathbf{A}_{ii}}\right). \tag{59}$$

This is in fact algebraically equivalent to the i^{th} component of the PSOR update and contains the PGS method as a special case of $\lambda = 1$. Observe that by (58) we are guaranteed a non increasing sequence of iterates by our relaxation method. The complete iterative method can be listed as

```
1 :  method PSOR(N, λ, x, A, b)
2 :    for k = 1 to N
3 :      for all i
4 :        r_i ← A_{i*} x + b_i
5 :        x_i ← max(0, x_i − λ r_i / A_{ii})
6 :      next i
7 :    next k
8 :  end method
```

where N is the maximum number of allowed iterations and λ is the relaxation parameter.

5.4 The Minimum Map Newton Method

Using the minimum map reformulation we have the root search problem where $\mathbf{H} : \mathbb{R}^n \mapsto \mathbb{R}^n$ is given by,

$$\mathbf{H}(\mathbf{x}) \equiv \begin{bmatrix} h(\mathbf{x}_1, \mathbf{y}_1) \\ \cdots \\ h(\mathbf{x}_n, \mathbf{y}_n) \end{bmatrix} = \mathbf{0}. \tag{60}$$

Recall $\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{b}$ so $\mathbf{y}_i = \mathbf{A}_{ii}\mathbf{x}_i + \mathbf{b}_i + \sum_{j \neq i} \mathbf{A}_{ij}\mathbf{x}_j$, thus

$$\mathbf{H}_i(\mathbf{x}) \equiv h(\mathbf{y}_i, \mathbf{x}_i), \tag{61a}$$

$$= \min\left(\left(\mathbf{A}_{ii}\mathbf{x}_i + \mathbf{b}_i + \sum_{j \neq i} \mathbf{A}_{ij}\mathbf{x}_j\right), \mathbf{x}_i\right). \tag{61b}$$

The idea is to use a Newton method to solve the nonsmooth equation (60). To do that we need to generalize the concept of derivative [Pan90]. The nonsmooth function $\mathbf{H}_i(x)$ is a *selection function* of the affine functions, $\mathbf{x}_i$ and $(\mathbf{A}\mathbf{x}+\mathbf{b})_i$. Further, each $\mathbf{H}_i$ is Lipschitz continuous and since each of the components fulfill this requirement, then so does $\mathbf{H}(\mathbf{x})$ [Sch94].

Definition 5.1 *Consider any vector function $\mathbf{F}:\mathbb{R}^n \mapsto \mathbb{R}^n$. If there exist a function $\mathbf{BF}(\mathbf{x},\Delta\mathbf{x})$ that is* positive homogeneous *in $\Delta\mathbf{x}$, that is, for any $\alpha \geq 0$*

$$\mathbf{BF}(\mathbf{x},\alpha\Delta\mathbf{x}) = \alpha\mathbf{BF}(\mathbf{x},\Delta\mathbf{x}), \tag{62}$$

such that the limit

$$\lim_{\Delta\mathbf{x}\to\mathbf{0}} \frac{\mathbf{F}(\mathbf{x}+\Delta\mathbf{x}) - \mathbf{F}(\mathbf{x}) - \mathbf{BF}(\mathbf{x},\Delta\mathbf{x})}{\parallel \Delta\mathbf{x} \parallel} = \mathbf{0} \tag{63}$$

exists, then we say that $\mathbf{F}$ is B-differentiable at $\mathbf{x}$, and the function $\mathbf{BF}(\mathbf{x},\cdot)$ is called the B-derivative.

Notice that since $\mathbf{H}(\mathbf{x})$ is Lipschitz and directionally differentiable, then it is B-differentiable. The B-derivative $\mathbf{BH}(\mathbf{x},\cdot)$ is continuous, piecewise linear, and positive homogeneous. Observe that the B-derivative as a function of $\mathbf{x}$ is a set-valued mapping. We will use the B-derivative to calculate a descent direction for the merit function,

$$\theta(\mathbf{x}) = \frac{1}{2}\mathbf{H}(\mathbf{x})^T\mathbf{H}(\mathbf{x}). \tag{64}$$

It is clear that a minimizer of (64) is a solution of equation (60). We use this B-derivative to formulate a linear subproblem whose solution will always provide a descent trajectory to (64). In fact, the biggest computational task for solving the nonsmooth and nonlinear system (60) is the solution of a large linear system of equations. This is similar to what Billups does to solve his nonsmooth system in [Bil95] and we repeat this same technique in this work.

The generalized Newton equation in the k^{th} iteration is

$$\mathbf{H}(\mathbf{x}^k) + \mathbf{BH}(\mathbf{x}^k,\Delta\mathbf{x}^k) = \mathbf{0}. \tag{65}$$

Each Newton iteration is finished by doing a correction of the previous iterate,

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \tau^k\Delta\mathbf{x}^k \tag{66}$$

where τ^k is called the step length and $\Delta\mathbf{x}^k$ is the Newton direction. The following theorems, see [Pan90, QS93, FK98, DLFK00], guarantee that $\Delta\mathbf{x}^k$ will always provide a descent direction for the the merit function $\theta(\mathbf{x})$.

Theorem 5.1 *Let $\mathbf{H}:\mathbb{R}^n \mapsto \mathbb{R}^n$ be B-differentiable, and let $\theta:\mathbb{R}^n \to \mathbb{R}$ be defined by*

$$\theta(\mathbf{x}) = \frac{1}{2}\mathbf{H}(\mathbf{x})^T\mathbf{H}(\mathbf{x}) \tag{67}$$

Then θ is B-differentiable and its directional derivative at $\mathbf{x}^k$ in direction $\Delta\mathbf{x}^k$ is

$$\mathbf{B}\theta(\mathbf{x}^k, \Delta\mathbf{x}^k) = \mathbf{H}(\mathbf{x}^k)^T\mathbf{B}\mathbf{H}(\mathbf{x}^k, \Delta\mathbf{x}^k). \tag{68}$$

Moreover, if (65) *holds then the directional derivative of θ is*

$$\mathbf{B}\theta(\mathbf{x}^k, \Delta\mathbf{x}^k) = -\mathbf{H}(\mathbf{x}^k)^T\mathbf{H}(\mathbf{x}^k) \tag{69}$$

Proof Since $\mathbf{H}$ is B-differentiable, (68) follows from the chain rule. Since $\mathbf{B}\theta(\mathbf{x}^k, \Delta\mathbf{x}^k) = \mathbf{H}(\mathbf{x}^k)^T\mathbf{B}\mathbf{H}(\mathbf{x}^k, \Delta\mathbf{x}^k)$, we have from the Newton equation (65) that $\mathbf{B}\theta(\mathbf{x}^k, \Delta\mathbf{x}^k) = -\mathbf{H}(\mathbf{x}^k)^T\mathbf{H}(\mathbf{x}^k)$.

Observe that a direct consequence of (69) is that any solution $\Delta\mathbf{x}^k$ of the generalized Newton equation (65) will always provide a descent direction to the merit function $\theta(\mathbf{x}^k)$. The following theorem shows that even if we solve (65) approximately, we can still generate a descent direction provided the residual is not too big.

Theorem 5.2 *Suppose we solve equation* (65) *approximately. That is, suppose that $\Delta\mathbf{x}^k$ satisfy the residue equation,*

$$\mathbf{r}^k = \mathbf{H}(\mathbf{x}^k) + \mathbf{B}\mathbf{H}(\mathbf{x}^k, \Delta\mathbf{x}^k), \tag{70}$$

and define the function $\theta(\mathbf{x})$ as above. Then, $\Delta\mathbf{x}^k$ will always provide a descent direction for $\theta(\mathbf{x}^k)$ provided

$$\| \mathbf{H}(\mathbf{x}^k) + \mathbf{B}\mathbf{H}(\mathbf{x}^k, \Delta\mathbf{x}^k) \| \leq \gamma \| \mathbf{H}(\mathbf{x}^k) \| \tag{71}$$

for some prescribed positive tolerance $\gamma < 1$.

Proof In order to show that the vector $\Delta\mathbf{x}$ provide a descent direction for $\theta(\mathbf{x})$ we need to prove that $\mathbf{B}\theta(\mathbf{x}^k, \Delta\mathbf{x}) < 0$. Suppose $\Delta\mathbf{x}^k$ satisfy (70), then $\mathbf{B}\mathbf{H}(\mathbf{x}^k, \Delta\mathbf{x}^k) = \mathbf{r}^k - \mathbf{H}(\mathbf{x}^k)$. We have that

$$\begin{aligned}
\mathbf{B}\theta(\mathbf{x}^k, \Delta\mathbf{x}^k) &= \mathbf{H}(\mathbf{x}^k)^T\mathbf{B}\mathbf{H}(x^k, \Delta\mathbf{x}^k), && \text{(72a)}\\
&= \mathbf{H}(\mathbf{x}^k)^T(\mathbf{r}^k - \mathbf{H}(\mathbf{x}^k)), && \text{(72b)}\\
&= -\mathbf{H}(\mathbf{x}^k)^T\mathbf{H}(\mathbf{x}^k) + \mathbf{H}(\mathbf{x}^k)^T\mathbf{r}^k. && \text{(72c)}
\end{aligned}$$

Now notice that $\mathbf{B}\theta(\mathbf{x}^k, \Delta\mathbf{x}^k) < 0$ if and only if

$$\mathbf{H}(\mathbf{x}^k)^T\mathbf{r}^k < \mathbf{H}(\mathbf{x}^k)^T\mathbf{H}(\mathbf{x}^k) \tag{73}$$

or $\mathbf{H}(\mathbf{x}^k)^T\mathbf{r}^k < \| \mathbf{H}(\mathbf{x}^k) \|^2$. This implies that $\mathbf{r}^k$ should lie inside a ball of radius at least $\| \mathbf{H}(\mathbf{x}^k) \|$, in other words $\| \mathbf{r}^k \| \leq \gamma \| \mathbf{H}(\mathbf{x}^k) \|$ for some $\gamma < 1$.

We will now present an efficient way of computing the B-derivative. Given the index i we have,

$$\mathbf{H}_i(\mathbf{x}) = \begin{cases} \mathbf{y}_i & \text{if } \mathbf{y}_i < \mathbf{x}_i \\ \mathbf{x}_i & \text{if } \mathbf{y}_i \geq \mathbf{x}_i \end{cases}. \tag{74}$$

Recall $\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{b}$. All of these are affine functions and from [Sch94] we can compute the B-derivative $\mathbf{B}\mathbf{H}_{ij} = \frac{\partial \mathbf{H}_i}{\partial \mathbf{x}_j}\Delta\mathbf{x}_j$ as follows

(1) If $\mathbf{y}_i < \mathbf{x}_i$ then

$$\frac{\partial \mathbf{H}_i}{\partial \mathbf{x}_j} = \mathbf{A}_{ij}. \tag{75}$$

(2) If $\mathbf{y}_i \geq \mathbf{x}_i$ then

$$\frac{\partial \mathbf{H}_i}{\partial \mathbf{x}_j} = \begin{cases} 1 & \text{if } j = i \\ 0 & \text{otherwise} \end{cases}. \tag{76}$$

We define two index sets corresponding to our choice of active selection functions,

$$\mathcal{A} \equiv \{i \mid \mathbf{y}_i < \mathbf{x}_i\} \text{ and } \mathcal{F} \equiv \{i \mid \mathbf{y}_i \geq \mathbf{x}_i\}. \tag{77}$$

Next we use a permutation of the indexes such that all variables with $i \in \mathcal{F}$ are shifted to the end. Hereby we have created the imaginary partitioning of the B-derivative,

$$\mathbf{BH}(\mathbf{x}^k, \Delta\mathbf{x}^k) = \begin{bmatrix} \mathbf{A}_{\mathcal{A}\mathcal{A}} & \mathbf{A}_{\mathcal{A}\mathcal{F}} \\ \mathbf{0} & \mathbf{I}_{\mathcal{F}\mathcal{F}} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{x}^k_{\mathcal{A}} \\ \Delta\mathbf{x}^k_{\mathcal{F}} \end{bmatrix}. \tag{78}$$

Notice this convenient block structure with $\mathbf{A}_{\mathcal{A}\mathcal{A}}$ a principal submatrix of $\mathbf{A}$. The matrix $\mathbf{I}_{\mathcal{F}\mathcal{F}}$ is an identify matrix of the same dimension as the $\mathcal{F}$.

If we use the blocked partitioning of our B-derivative from (78) then the corresponding permuted version of the Newton equation (65) is

$$\begin{bmatrix} \mathbf{A}_{\mathcal{A}\mathcal{A}} & \mathbf{A}_{\mathcal{A}\mathcal{F}} \\ \mathbf{0} & \mathbf{I}_{\mathcal{F}\mathcal{F}} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{x}^k_{\mathcal{A}} \\ \Delta\mathbf{x}^k_{\mathcal{F}} \end{bmatrix} = - \begin{bmatrix} \mathbf{H}_{\mathcal{A}}(\mathbf{x}^k) \\ \mathbf{H}_{\mathcal{F}}(\mathbf{x}^k) \end{bmatrix}. \tag{79}$$

Observe that this can be trivially reduced to

$$\mathbf{A}_{\mathcal{A}\mathcal{A}}\Delta\mathbf{x}^k_{\mathcal{A}} = \mathbf{A}_{\mathcal{A}\mathcal{F}}\mathbf{H}_{\mathcal{F}} - \mathbf{H}_{\mathcal{A}}. \tag{80}$$

Our problem is reduced to a potentially smaller linear system in $\Delta\mathbf{x}^k_{\mathcal{A}}$. Whether an exact solution can be found for this reduced system depends on the matrix properties of the original matrix $\mathbf{A}$. For physics-based animation this matrix could be symmetric PSD implying that the reduced matrix could inherit these properties and one might end up with a singular system to solve. The good news is that one does not need an accurate solution to guarantee a descent as we proved previously. In practice we have found GMRES to be suitable as a general purpose choice.

Due to the connection of the generalized Newton method with the classical Newton method, global convergence is unlikely if we start with an arbitrary iterate $\mathbf{x}^1$. To remedy this we perform an Armijo type line search on our merit function $\theta(\cdot)$. The ideal choice for a step length τ^k would be a global minimizer for the scalar function $\psi(\tau) = \theta(\mathbf{x}_\tau)$ where $\mathbf{x}_\tau = \mathbf{x}^k + \tau\Delta\mathbf{x}^k$. In practice this could be expensive to compute, requiring too many evaluations of $\theta(\cdot)$ and possibly $\mathbf{B}\theta(\cdot,\cdot)$. The Armijo condition stipulates that the reduction in $\psi(\tau)$ should be proportional to both the step length τ^k and the directional derivative $\nabla\psi(0) = \mathbf{B}\theta(\mathbf{x}^k, \Delta\mathbf{x}^k)$. For a sufficient decrease parameter $\alpha \in (0, 1)$ we state this as

$$\psi(\tau^k) \leq \psi(0) + \alpha\tau^k\nabla\psi(0). \tag{81}$$

To rule out unacceptably short steps we introduce a second condition, called the *curvature condition*,

$$\nabla\psi(\tau^k) \geq \sigma\nabla\psi(0) \tag{82}$$

where the curvature parameter is selected such that $\sigma \in (\alpha,\, 1)$. Conditions (81) and (82) are known collectively as the *Wolfe conditions*. If the line search method chooses its candidate step length appropriately, by using a *back-tracking* approach, we can eliminate the extra condition (82). Now the Armijo condition implies to find the largest $h \in \mathbf{Z}_0$ such that

$$\psi(\tau^k) \leq \psi(0) + \alpha\tau^k\nabla\psi(0) \tag{83}$$

where $\tau^k = \beta^h\tau^0$, $\tau^0 = 1$, and the step-reduction parameter $\alpha < \beta < 1$. In Nocedal and Wright it reads that α is often quite small, say $\alpha = 10^{-4}$ and $\beta = \frac{1}{2}$ is often used [NW99].

In practice we have observed that the minimum map Newton method may converge to a local minimum corresponding to an infeasible iterate. To remedy this we apply a projected Armijo back-tracking line search. This means we project the line search iterate $\mathbf{x}_\tau = \max(\mathbf{0}, \mathbf{x}^k + \tau\Delta\mathbf{x}^k)$ before computing the value of the merit function $\psi(\tau) = \theta(\mathbf{x}_\tau)$.

```
1 :  method projected-line-search
2 :    (ψ0, ∇ψ0) ← (θ(x^k), Bθ(x^k, Δx^k))
3 :    τ ← 1
4 :    while forever
5 :      x_τ ← max(0, x^k + τΔx^k)
6 :      ψ_τ ← θ(x_τ)
7 :      if ψ_τ ≤ ψ0 + ατ∇ψ0 then
8 :        return τ
9 :      end if
10 :     τ ← βτ
11 :   end while
12 :  end method
```

The back-tracking line search method we have outlined is general and could be used with any merit function. In rare cases one may experience that τ becomes too small. Thus, it may be beneficial to add an extra stop criterion after line 9 testing if $\tau < \delta$ where $0 < \delta \ll 1$ is some user-specified tolerance. If the test passes one simply returns the current value of τ as the τ^k value.

Many have commented that globalizing the minimum map Newton method is difficult. As far as we know the projected back tracking line search has not been reported in the literature. Instead Levenberg-Marquard style search directions are chosen with an occasional gradient descent direction [FK98, DLFK00].

We now combine all the ingredients of the minimum map Newton method into pseudo code.

```
1 :  method minimum-map-Newton
2 :   while forever
3 :    y^k ← Ax^k + b
4 :    H^k ← min(y^k, x^k)
5 :    A ← {i | y_i < x_i}
6 :    F ← {i | y_i ≥ x_i}
7 :    Δx^k_F ← −H^k_F
8 :    solve A_AA Δx^k_A = A_AF H^k_F − H^k_A
9 :    τ^k ← projected-line-search(...)
10 :   x^{k+1} ← x^k + τ^k Δx^k
11 :   if converged then
12 :    return x^{k+1}
13 :   end
14 :   k ← k + 1
15 :  end while
16 : end method
```

We will discuss possible stopping criteria later in Section 6.

5.5 The Fischer–Newton Method

We will introduce another Newton method based on [Fis92]. Many have investigated this formulation[FK98, KK98, DLFK00, CK00]. We use the reformulation

$$\mathbf{F}(\mathbf{x}) = \mathbf{F}(\mathbf{x}, \mathbf{y}) = \begin{bmatrix} \phi(\mathbf{x}_1, \mathbf{y}_1) \\ \vdots \\ \phi(\mathbf{x}_n, \mathbf{y}_n) \end{bmatrix} = \mathbf{0}. \tag{84}$$

This is a nonsmooth root search problem solved using a generalized Newton method. In an iterative fashion one solves the generalized Newton equation

$$\mathbf{J}\Delta\mathbf{x}^k = -\mathbf{F}(\mathbf{x}^k) \tag{85}$$

for the Newton direction $\Delta\mathbf{x}^k$. Here $\mathbf{J} \in \partial\mathbf{F}(\mathbf{x}^k)$ is any member from the generalized Jacobian $\partial F(\mathbf{x}^k)$. Then the Newton update yields

$$\mathbf{x}^{k+1} = \mathbf{x}^k + \tau^k \Delta\mathbf{x}^k \tag{86}$$

where τ^k is the step length of the k^{th} iteration.

Definition 5.2 *Given* $\mathbf{F}$ *and let* $\mathcal{D} \subset \mathbb{R}^n$ *be the set of all* $\mathbf{x} \in \mathbb{R}^n$ *where* $\mathbf{F}$ *is continuously differentiable. Further, assume* $\mathbf{F}$ *is Lipschitz continuous at* $\mathbf{x}$ *then the B–subdifferential of* $\mathbf{F}$ *at* $\mathbf{x}$ *is defined as*

$$\partial_B\mathbf{F}(\mathbf{x}) \equiv \left\{ \mathbf{J} \;\middle|\; \exists \mathbf{x}_k \subset \mathcal{D} \;\wedge\; \lim_{\mathbf{x}_k \to \mathbf{x}} \frac{\partial\mathbf{F}(\mathbf{x}_k)}{\partial\mathbf{x}} = \mathbf{J} \right\}. \tag{87}$$

Definition 5.3 *Clarke's generalized Jacobian of* $\mathbf{F}$ *at* $\mathbf{x}$ *is defined as the convex hull of the B–subdifferential [Cla90],*

$$\partial \mathbf{F}(\mathbf{x}) \equiv \boldsymbol{co}\left(\partial_B \mathbf{F}(\mathbf{x})\right). \tag{88}$$

Let us look at an example. Consider the Euclidean norm $e : \mathbb{R}^2 \mapsto \mathbb{R}$ defined as

$$e(\mathbf{z}) \equiv \| \mathbf{z} \| = \sqrt{\mathbf{z}^T \mathbf{z}} \tag{89}$$

Then, for $\mathbf{z} \in \mathbb{R}^2 \setminus \{\mathbf{0}\}$ we have

$$\partial e(\mathbf{z}) = \partial_B e(\mathbf{z}) = \frac{\partial e(\mathbf{z})}{\partial \mathbf{z}} = \frac{\mathbf{z}^T}{\| \mathbf{z} \|} \quad \forall \mathbf{z} \neq \mathbf{0}. \tag{90}$$

For $\mathbf{z} = \mathbf{0}$ we have

$$\partial_B e(\mathbf{0}) = \{\mathbf{v}^T \mid \mathbf{v} \in \mathbb{R}^2 \ \wedge \ \| \mathbf{v} \| = 1\}, \tag{91a}$$

$$\partial e(\mathbf{0}) = \{\mathbf{v}^T \mid \mathbf{v} \in \mathbb{R}^2 \ \wedge \ \| \mathbf{v} \| \leq 1\}. \tag{91b}$$

We now have most pieces to deal with the generalized Jacobian of the Fischer-Burmeister function. For $\mathbf{z} = \begin{bmatrix} x & y \end{bmatrix}^T \in \mathbb{R}^2$ we may write the Fischer-Burmeister function as

$$\phi(x, y) \equiv \phi(\mathbf{z}) \equiv e(\mathbf{z}) - g(\mathbf{z}) \tag{92}$$

where $g(\mathbf{z}) = \left(\begin{bmatrix} 1 & 1 \end{bmatrix}^T \mathbf{z} \right)$. From this we find

$$\partial_B \phi(\mathbf{z}) = \partial_B e(\mathbf{z}) - \frac{\partial g(\mathbf{z})}{\partial \mathbf{z}}, \tag{93a}$$

$$\partial \phi(\mathbf{z}) = \partial e(\mathbf{z}) - \frac{\partial g(\mathbf{z})}{\partial \mathbf{z}}. \tag{93b}$$

Hence for $\mathbf{z} \neq \mathbf{0}$,

$$\partial \phi(\mathbf{z}) = \partial_B \phi(\mathbf{z}) = \left\{ \frac{\mathbf{z}^T}{\| \mathbf{z} \|} - \begin{bmatrix} 1 & 1 \end{bmatrix}^T \right\} \tag{94}$$

and

$$\partial_B \phi(\mathbf{0}) = \{\mathbf{v}^T - \begin{bmatrix} 1 & 1 \end{bmatrix}^T \mid \mathbf{v} \in \mathbb{R}^2 \ \wedge \ \| \mathbf{v} \| = 1\}, \tag{95a}$$

$$\partial \phi(\mathbf{0}) = \{\mathbf{v}^T - \begin{bmatrix} 1 & 1 \end{bmatrix}^T \mid \mathbf{v} \in \mathbb{R}^2 \ \wedge \ \| \mathbf{v} \| \leq 1\}. \tag{95b}$$

Having studied the one-dimensional case we can now move on to the higher-dimensional case.

Theorem 5.3 *The Generalized Jacobian of the Fischer-Burmeister reformulation can be written as*

$$\partial \mathbf{F}(\mathbf{x}) \equiv \mathbf{D}_p(\mathbf{x}) + \mathbf{D}_q(\mathbf{x}) \mathbf{A} \tag{96}$$

where $\mathbf{D}_p(\mathbf{x}) = \boldsymbol{diag}\left(p_1(\mathbf{x}), \ldots, p_n(\mathbf{x})\right)$ *and* $\mathbf{D}_q(\mathbf{x}) = \boldsymbol{diag}\left(q_1(\mathbf{x}), \ldots, q_n(\mathbf{x})\right)$ *are diagonal matrices. If* $\mathbf{y}_i \neq 0$ *or* $\mathbf{x}_i \neq 0$ *then*

$$p_i(\mathbf{x}) = \frac{\mathbf{x}_i}{\sqrt{\mathbf{x}_i^2 + \mathbf{y}_i^2}} - 1, \tag{97a}$$

$$q_i(\mathbf{x}) = \frac{\mathbf{y}_i}{\sqrt{\mathbf{x}_i^2 + \mathbf{y}_i^2}} - 1, \tag{97b}$$

else if $\mathbf{y}_i = \mathbf{x}_i = 0$ *then*

$$p_i(\mathbf{x}) = a_i - 1, \tag{98a}$$

$$q_i(\mathbf{x}) = b_i - 1 \tag{98b}$$

for any $a_i, b_i \in \mathbb{R}$ *such that* $\| \begin{bmatrix} a_i & b_i \end{bmatrix}^T \| \leq 1$

Proof Assume $\mathbf{y}_i \neq 0$ or $\mathbf{x}_i \neq 0$ then the differential is

$$d\mathbf{F}_i(\mathbf{x}, \mathbf{y}) = d\left(\left(\mathbf{x}_i^2 + \mathbf{y}_i^2\right)^{\frac{1}{2}}\right) - d\left(\mathbf{x}_i + \mathbf{y}_i\right) \tag{99}$$

By the chain rule

$$d\mathbf{F}_i(\mathbf{x}, \mathbf{y}) = \frac{1}{2}\left(\mathbf{x}_i^2 + \mathbf{y}_i^2\right)^{-\frac{1}{2}} d\left(\mathbf{x}_i^2 + \mathbf{y}_i^2\right) - d\mathbf{x}_i - d\mathbf{y}_i, \tag{100a}$$

$$= \frac{\mathbf{x}_i d\mathbf{x}_i + \mathbf{y}_i d\mathbf{y}_i}{\sqrt{\mathbf{x}_i^2 + \mathbf{y}_i^2}} - d\mathbf{x}_i - d\mathbf{y}_i, \tag{100b}$$

$$= \begin{bmatrix} \underbrace{\left(\frac{\mathbf{x}_i}{\sqrt{\mathbf{x}_i^2 + \mathbf{y}_i^2}} - 1\right)}_{p_i(\mathbf{x})} & \underbrace{\left(\frac{\mathbf{y}_i}{\sqrt{\mathbf{x}_i^2 + \mathbf{y}_i^2}} - 1\right)}_{q_i(\mathbf{x})} \end{bmatrix} \begin{bmatrix} d\mathbf{x}_i \\ d\mathbf{y}_i \end{bmatrix}. \tag{100c}$$

Finally $d\mathbf{y} = \mathbf{A} d\mathbf{x}$, so $d\mathbf{y}_i = \mathbf{A}_{i*} d\mathbf{x}$ by substitution

$$d\mathbf{F}_i(\mathbf{x}, \mathbf{y}) = \underbrace{\left(p_i(\mathbf{x})\mathbf{e}_i^T + q_i(\mathbf{x})\mathbf{A}_{i*}\right)}_{\partial \mathbf{F}_i(\mathbf{x})} d\mathbf{x}. \tag{101}$$

The case $\mathbf{x}_i = \mathbf{y}_i = 0$ follows from the previous examples.

The next problem we are facing is how to solve the generalized Newton equation. The issue is that we need to pick one element $\mathbf{J}$ from $\partial \mathbf{F}(\mathbf{x}^k)$. We have explored four strategies.

Random One strategy may be to pick a random value for all $\mathbf{a}_i$ and $\mathbf{b}_i$ where $\mathbf{x}_i = \mathbf{y}_i = 0$.

Zero Instead of random values one may pick $\mathbf{a}_i = \mathbf{b}_i = 0$.

Perturbation Another strategy may be to perturb the problem slightly whenever $\mathbf{x}_i = \mathbf{y}_i = 0$. For instance given a user-specified tolerance $0 < \varepsilon \ll 1$ one could use $\mathbf{x}_i' = \varepsilon$ in-place of $\mathbf{x}_i$ when evaluating the generalized Jacobian.

Approximation One could exploit the cases where the Newton equation is solved with an iterative method such as preconditioned conjugate gradient (PCG) method or generalized minimum residual (GMRES) method. Then one only need to compute matrix vector products $\mathbf{Jp}$ for some given search direction vector $\mathbf{p}$. By definition of directional derivative

$$\mathbf{Jp} = \lim_{h \to 0^+} \frac{\mathbf{F}(\mathbf{x} + h\mathbf{p}) - \mathbf{F}(\mathbf{x})}{h}. \tag{102}$$

This means we can numerically approximate $\mathbf{Jp}$ using finite differences.

To globalize the Fischer–Newton method we apply the same projected Armijo back-tracking line search explained previously. We redefine the natural merit function to be,

$$\theta(\mathbf{x}) \equiv \frac{1}{2}\mathbf{F}(\mathbf{x})^T\mathbf{F}(\mathbf{x}). \tag{103}$$

Assuming we have a solution to $\mathbf{J}\Delta\mathbf{x}^k = -\mathbf{F}(\mathbf{x}^k)$ then by the chain rule we find the directional derivative of the merit function to be,

$$\nabla\theta(\mathbf{x}^k)^T\Delta\mathbf{x}^k = \mathbf{F}(\mathbf{x})^T\mathbf{J}\Delta\mathbf{x}^k. \tag{104}$$

These are the modifications needed in the projected back-tracking line search method. Observe that an accurate solution for $\Delta\mathbf{x}^k$ will always result in a descent direction as $\nabla\theta(\mathbf{x}^k)^T\Delta\mathbf{x}^k = -2\theta(\mathbf{x}^k) < 0$. Defining the residual vector as $\mathbf{r}^k = \mathbf{F}(\mathbf{x}^k) + \mathbf{J}\Delta\mathbf{x}^k$ and following the same recipe from our previous proof in Section 5.4, one may show that if

$$\| \mathbf{r}^k \| < \gamma \| \mathbf{F}(\mathbf{x}^k) \| \tag{105}$$

for some prescribed $0 < \gamma < 1$ then $\Delta\mathbf{x}^k$ will be a descent direction for the merit function. Observe that this result is useful to determine a sufficient stopping threshold for an iterative linear system method that will guarantee a descent direction.

The pseudo code of the Fischer–Newton method is,

```
1 :  method Fischer-Newton
2 :   while forever
3 :    solve J Δx^k = −F^k
4 :    τ^k ← projected-line-search(...)
5 :    x^{k+1} ← x^k + τ^k Δx^k
6 :    if converged then
7 :     return x^{k+1}
8 :    end
9 :    k ← k + 1
10 :   end while
11 :  end method
```

We will discuss possible stopping criteria later in Section 6.

6 Tips, Tricks and Implementation Hacks

The Newton equations for both Newton methods can be solved using an iterative linear system method. We have successfully applied the two Krylov subspace methods PCG or GMRES [Saa03]. GMRES is more general than PCG and can be used for any non-singular matrix whereas PCG requires the matrix to be symmetric PD. PCG can not be used for the full Newton equation in case of the minimum map reformulation. However, for the Shur reduced system it may be possible if the principal submatrix is symmetric PD. GMRES is more general purpose and one incurs an extra storage cost for this.

The iterative linear system methods are an advantage in combination with the finite difference *approximation* strategy that we introduced for the Fischer–Newton method. The same trick could be applied for the minimum map Newton method. The advantage from this numerical approximation is an overall numerical method that does not have to assemble the global $\mathbf{A}$-matrix. Instead it can work with this matrix in an implicit form or using some known factorization of the matrix. In particular for interactive rigid body dynamics this is an advantage as a global matrix-free method holds the possibility of linar computation time scaling rather than quadratic in the number of variables. Often the storage will only be linear for a factorization whereas the global matrix could be dense in the worst case and require quadratic storage complexity. For fluid problems one would often not assemble the global matrix but rather use the finite difference stencils on the regular fluid grid to implicitly compute matrix-vector products. Thus, for fluid problems iterative linear system methods should be used. For the Newton methods one can in fact use PCG. In our implementation we use GMRES to keep our Newton methods more general.

We have not experimented with preconditioners. It should be noted that PCG can deal with PSD matrix if for instance an incomplete Cholesky preconditioner is applied. GMRES would most likely benefit from the same type of preconditioner as the PCG method.

Newton methods can benefit if one uses a good starting iterate. For the Newton methods that we have derived one would usually be content with using $\mathbf{x} = \mathbf{0}$ as the starting iterate. For problem cases where PGS applies one can create a hybrid solution taking advantage of the robustness and low iteration cost of PGS to quickly within a few iterations compute a starting iterate for a Newton method.

All the iterative methods we have introduced would at some point require stopping criteria to test if a method has converged or if some unrecoverable situation has been encountered. To monitor this convergence process a numerical method would use a merit function. We already saw two definitions of merit functions for the Newton methods. For the PGS the QP reformulation may serve as the definition of the merit function. However, one may want to use the modified merit function definition

$$\theta(\mathbf{x}) = \mathbf{x}^T \left|\mathbf{A}\mathbf{x} + \mathbf{b}\right|. \tag{106}$$

This has the benefit that the merit function is bounded from below by zero and

the drawback that it does not work for $\mathbf{x} = \mathbf{0}$. It is often a good principle not to use only one stopping criterion but rather use a combination of them. For instance an absolute stopping criterion would be

$$\theta(\mathbf{x}^{k+1}) < \varepsilon_{\mathbf{abs}} \tag{107}$$

for some user-specified tolerance $0 < \varepsilon_{\mathbf{abs}} \ll 1$. In some cases convergence may be too slow. In these cases a relative convergence test is convenient,

$$\left|\theta(\mathbf{x}^{k+1}) - \theta(\mathbf{x}^{k})\right| < \varepsilon_{\mathbf{rel}} \left|\theta(\mathbf{x}^{k})\right| \tag{108}$$

for some user-specified tolerance $0 < \varepsilon_{\mathbf{rel}} \ll 1$. A simple guard against the number of iterations exceeding a prescribed maximum helps avoid infinite looping. A stagnation test helps identifying numerical problems in the iterative values

$$\max_i \left|\mathbf{x}_i^{k+1} - \mathbf{x}_i^{k}\right| < \varepsilon_{\mathbf{stg}} \tag{109}$$

for some user-specified tolerance $0 < \varepsilon_{\mathbf{stg}} \ll 1$. This test usually only works "numerically" well for numbers close to one. A rescaled version may be better for large numbers.

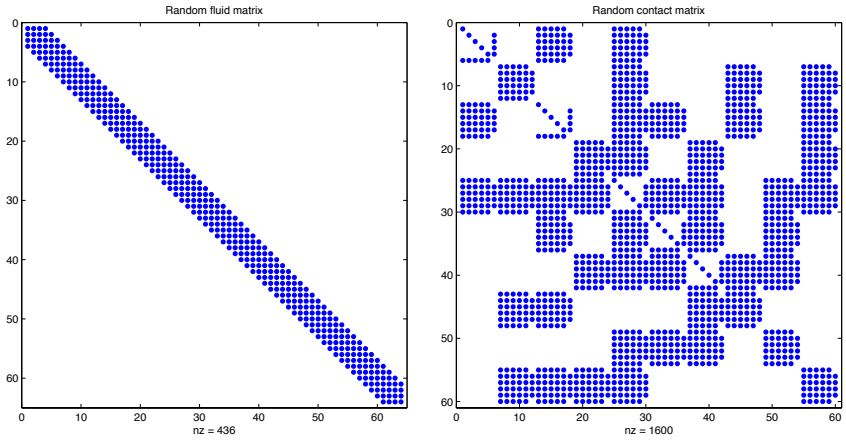
Besides the above stopping criteria one may verify numerical properties. For instance for the Newton type methods it can be helpful to verify that the Newton direction is a descent direction. Global convergence of the Newton methods often implies that they converge to an iterate with a zero gradient. This is not the same as having found a global minimizer. Thus, it may be insightful to test if the gradient of the merit function is close to zero and halt if this is the case.

What should one do in case one does not converge to something meaningful? In off-line simulations one may have the luxury to restart a Newton method with a new starting iterate and hope that the "bad" behavior will be avoided. This may not be the best way to spend computing time or it may even be impossible in an interactive simulator. One remedy is to fall back on using the gradient direction as the search direction whenever one fails to have a well defined Newton direction to work with.

7 Convergence, Performance and Robustness Experiments

We have implemented all the numerical methods in Matlab (R2010a) [Erl11] and run our experiments on a MacBook with 2.4 GHz Intel Core 2 Duo, and 4 GB RAM on Mac OSX 10.6.8. All tests took approximately 100 computing hours.

To quickly generate a large number of test runs and automate our experiments the Matlab suite contains a small small suite of methods that can generate synthetic "fake" fluid or contact LCPs with the properties we discussed in Section 4. Fig. 1 shows two examples of generated matrices for the LCPs. This



(a) (b)

Figure 1: Fill patterns of a fluid matrix (a) and a contact matrix (b). Observe that the matrices are extremly sparse and that the fluid matrix is symmetric whereas the contact matrix is non-symmetric.

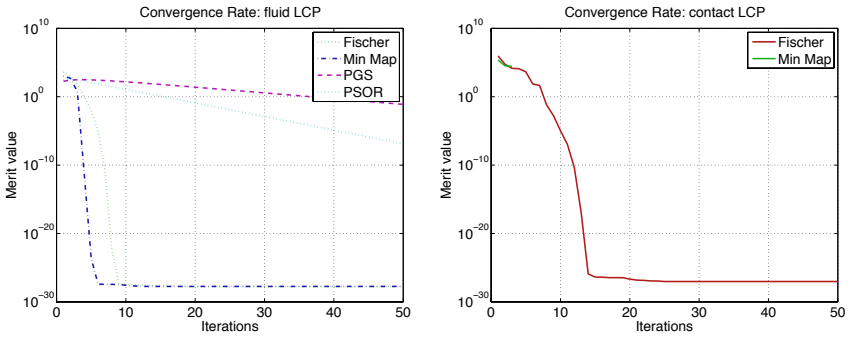
has the added benefit of being able to quickly re-run experiments under varying parameters and offers a great deal of control.

Our synthetic tests are no substitute for real world problems and only serve to demonstrate the inherent convergence properties. As a remark we note that all numerical methods have been tested on RPI's Matlab rigid body simulator [rpi] and compared against PATH [FM99]. Fischer–Newton method rivals PATH robustness but scales better due to the iterative sub-solver. PATH seems slightly more robust due to its non-monotone line-search method.

We examine the convergence rate of the iterative methods using a 1000-variable fluid problem and 300-variable contact problem. The problem sizes are limited by the Matlab programming environment. For the PSOR and PGS methods we use the modified merit function (106). The Fischer–Newton and minimum map Newton method uses their natural merit functions. Our results are shown in Fig. 2. As expected we observe linear convergence rate for PSOR and PGS while Newton methods show quadratic convergence rate.

Due to a non-singular Newton equation matrix the minimum map Newton method does not work on contact problems. It gives up after a few iterations where it gets into a non-descent iterate. This leaves only the Fischer–Newton method as a real alternative for contact problems. For fluid problems the minimum map Newton method finished in lower iterations. Finally we observe that PSOR using $\lambda = 1.4$ converges faster than PGS.

Next we will examine how the different Newton equation strategies for the Fischer–Newton method affect the overall convergence behavior. For this we have generated a fluid LCP corresponding to 1000 variables and a contact LCP



(a) (b)

Figure 2: Convergence rates for a fluid (a) and a contact problem (b). For the fluid problem we observe high accuracy of the Newton methods within few iterations. For the contact problem only the Fischer–Newton method works.

with 180 variables. Our results are shown in Fig. 3. For contact problems we have observed that the *perturbation* strategy at times uses fewer iterations than the other strategies. For fluid problems all strategies appear similar in convergence behavior. The *approximation* strategy always result in a less accurate solution than the other strategies.

We performed a parameter study of PSOR for a 1000 variable fluid problem. Here we plotted convergence rate for various relaxation parameter values as shown in Fig. 4. Our results show that a value of $\lambda = 1.4$ seems to work best.

For all iterative methods we have measured the wall clock time for 10 iterations. The results of our performance measurements are shown in Fig. 5. Not surprisingly we observe the cubic scaling of the Lemke method. This will quickly make it intractable for large problems [3]. In case of the fluid problem we found the Fischer–Newton method to scale worse than linear this is unexpected as we use GMRES for solving the Newton equation. The explanation of the behavior is that in our implementation we are always assembling the Jacobian matrix $\mathbf{J}$. We do this to add extra safe guards in our implementation against non descent iterates. If we apply dense matrices then the assembly of the Jacobian will scale quadratically if sparse matrices are used the assembly will scale in the number of non zeros. Theoretically, if one omitted the safe guards and used the *approximation* strategy then the Fischer–Newton iteration should scale linear.

We examined the robustness of our implemented methods. For this we have generated 100 fluid problems with 1000 variables and 100 contact problems with 180 variables. In all cases we used a relative tolerance of 10^{-6}, an absolute tolerance of 10^{-3}, and a maximum upper limit of 100 iterations. For each invocation we recorded the final state as being *relative* convergence, *absolute*

[3] A multilevel method [Erl11] is shown. Due to space considerations and poor behavior we have omitted all details about this method in these course notes.

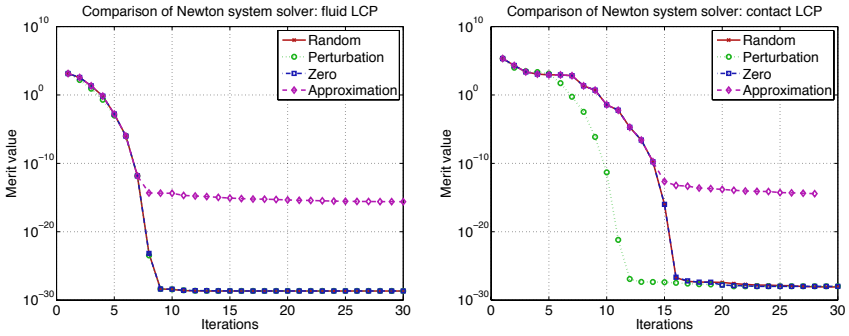


Figure 3: Convergence rates of Fischer–Newton method using different strategies. A fluid (a) and a contact (b) problem are displayed. Observe that the *perturbation* strategy works a little better for the contact case and that the *approximation* strategy is less accurate.

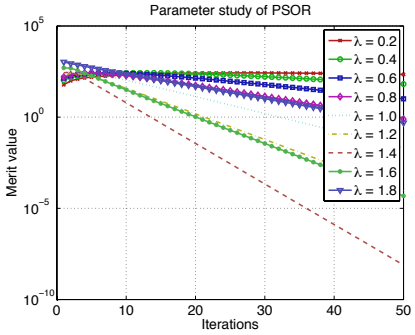
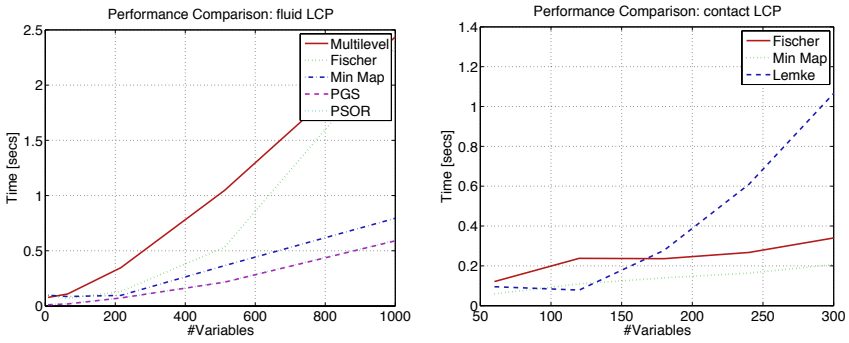


Figure 4: Parameter study of PSOR. Notice that a value of approximately 1.4 seems to be best.



(a) (b)

Figure 5: Performance measurements for fluid problems (a) and contact problems (b).

Table 1: Robustness for 100 fluid and contact problems.

(a) Final state on fluid problems

	Relative	Absolute
Fischer	0	100
Min. Map.	0	100
PGS	0	100
PSOR	0	100

(b) Final state on contact problems

	Absolute	Non-descent
Fischer	100	0
Min. Map.	0	100

convergence, *stagnation* of iterates, *local minimum* iterate, *non descent* iterate, or *max iteration* limit reached. Table 1 displays our results. The minimum map Newton method does not work at all for contact problems. We observe that the rest of the methods are robust for this test setup.

We investigated the number of iterations required for absolute and relative convergence. The results are shown in Table 2. We observe a low standard deviation for the Newton type methods. This suggest that one a priori can determine suitable maximum limits for these types of methods. PSOR is on average twice as fast as PGS. The iterations of PGS vary wildly. This implies that PGS does not have predictable performance if accurate solutions are wanted.

Newton methods can benefit from a good initial starting iterate. To illustrate the impact of this we have generated 100 dense PD problems with 100 variables. We solved the problems using a zero valued starting iterate and a starting iterate obtained from 10 iterations of PGS. In the tests we used a relative tolerance of 10^{-6}, an absolute tolerance of 10^{-2}, and a maximum upper limit of 30 iterations for the Newton methods. Fig. 6 summarizes our findings.

To reach absolute convergence we observe that warm starting reduces the number of iterations to less than or equal to half of the number of iteration without warm starting. Considering the computational cost of PGS this is a

Table 2: Statistics on number of iterations.

(a) Absolute convergence contact problem

Method	Mean	Min	Max	Std
Fischer	14.54	9.00	26.00	2.79

(b) Absolute convergence fluid problem

Method	Mean	Min	Max	Std
Fischer	5.00	5.00	5.00	0.00
Min map	4.07	3.00	5.00	0.29
PGS	49.38	9.00	83.00	17.48
PSOR	25.29	16.00	35.00	3.77

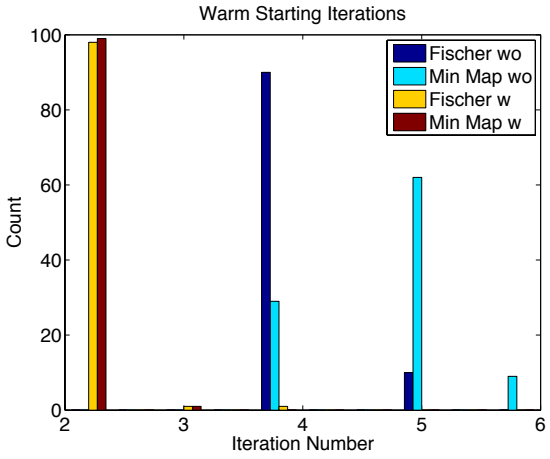


Figure 6: The number of iterations used to reach absolute convergence with (w) and without (wo) warm starting using PGS.

good tradeoff.

We have tested the numerical methods ability to deal with over determined systems. This is numerically equivalent to zero eigenvalues. For an increasing ratio of zero eigenvalues we have generated 100 dense PSD problems. For all cases we used a relative tolerance of 10^{-4}, absolute tolerance of 10^{-2} and a maximum iteration bound of 100. In our initial trial runs we have observed that the *approximate* strategy works better for PSD problems whereas the other strategies behave similar to the minimum map Newton method. We therefore applied the *approximation* strategy for this experiment. Due to space considerations we omit showing detailed histograms of the final solver states.

We observe that overall Fischer–Newton method seems better at reaching relative convergence whereas the minimum map Newton is less successful. For large ratios of zero eigenvalues we clearly see that both Newton methods get into trouble. A high number of zero eigenvalues cause ill conditioning or even singularity of the Newton equations and result in a local minimum or a non descent iterate. In all test runs the PGS and PSOR methods ends up reaching their maximum iteration limit. This is a little misleading as it does not mean they end up with a bad iterate rather it implies they converge slowly.

8 Discussion and Future Work

We have shown the LCP model for contact forces has a coefficient matrix that is non-symmetric and has a zero-diagonal block. This limits the type of numerical methods that can be used to either pivoting methods like Lemke or Keller, or Newton methods like the Fischer–Newton method. There exist general purpose Newton methods for LCPs such as PATH from CPNET [FM99]. PATH is a general solver and can solve nonlinear complementarity problems (NCPs) whereas the Newton-based methods we present are more lean and mean and tailored for the specific LCPs that is encountered in physics-based animation. For instance PATH scales quadratically in the number of unknowns as it requires the entire $\mathbf{A}$-matrix of the LCP. The Fischer–Newton method we have outlined can exploit factorizations or numerically approximate the matrix-vector products and achieve better scaling than PATH.

If one dislikes pivoting or Newton methods then one may apply the idea of staggering [KSJP08] to the LCP contact force model (27). In terms of our notation this would require us to solve the two coupled LCPs,

$$0 \leq \mathbf{B}_{nn}\lambda_n + (\mathbf{b}_n + \mathbf{B}_{nt}\lambda_t) \quad \perp \quad \lambda_n \geq 0, \tag{110a}$$

$$0 \leq \begin{bmatrix} \mathbf{B}_{tt} & \mathbf{e} \\ -\mathbf{e}^T & 0 \end{bmatrix} \begin{bmatrix} \lambda_t \\ \beta \end{bmatrix} + \begin{bmatrix} \mathbf{b}_t + \mathbf{B}_{tn}\lambda_n \\ \mu\lambda_n \end{bmatrix} \quad \perp \quad \begin{bmatrix} \lambda_t \\ \beta \end{bmatrix} \geq \mathbf{0}. \tag{110b}$$

Taking a staggered approach one solves the top-most LCP first (normal force problem) and then the bottom-most LCP second (the friction force problem) and continues iteratively until a fixed-point is reached. Observe that the normal force problem has a symmetric PSD coefficient matrix $\mathbf{B}_{nn}$ making QP reformulations possible whereas the frictional problem has a non-symmetric matrix.

One may exploit a QP reformulation anyway, because the friction LCP corresponds to the first-order optimality conditions of the QP problem

$$\lambda_t^* = \arg\min \ \frac{1}{2}\lambda_t^T \mathbf{B}_{tt}\lambda_t + \mathbf{c}_t^T \lambda_t \tag{111}$$

subject to

$$\lambda_t \geq \mathbf{0} \quad \text{and} \quad c_n - \mathbf{e}^T \lambda_t \geq 0, \tag{112}$$

where $c_n = \mu\lambda_n$ and $\mathbf{c}_t = \mathbf{b}_t + \mathbf{B}_{tn}\lambda_n$. Thus, any convex QP method can be used to solve for the normal and friction forces and one is guaranteed to find a solution for each subproblem. Whether the sequence of sub QP problems converge to a fixed point is not obvious.

All the methods we have covered could in principle be applicable for the fluid LCP problem. The splitting/QP-based PGS methods are straightforward to apply even in cases were one does not have the global matrix $\mathbf{A}$. For the Newton type methods one could evaluate the finite difference approximations directly on the regular grid to avoid building the $\mathbf{A}$-matrix. Given the properties of the $\mathbf{A}$-matrix these methods should be able to use PCG for solving the Newton equations. This could result in fast Newton methods with quadratic convergence rates and computing time that scales linear in the number of fluid grid nodes.

Table 3 summarizes properties of the numerical methods we have covered in these notes. Observe that the column "$\mathbf{A}$-matrix properties" is limited to the problem classes we used in our discussions.

Table 3: Numerical properties of numerical methods. The table shows worst-case complexities under the assumption that direct methods are used for linear systems. Convergence properties are not listed for pivoting methods as these give exact solutions in one single iteration. Time complexity refers to per iteration cost for the iterative methods and total cost for pivoting methods.

Method	Type	$\mathbf{A}$-Matrix Properties	Time Complexity	Storage Complexity	Convergence Rate	Global Convergence
(splitting) PGS	Iterative	Nonzero diagonal	$\mathcal{O}(n)$	$\mathcal{O}(n)$	Linear	No
(QP) PGS/PSOR	Iterative	Symmetric PSD	$\mathcal{O}(n)$	$\mathcal{O}(n)$	Linear	Yes
Dantzig	Pivoting	Symmetric PD	$\mathcal{O}(n^4)$	$\mathcal{O}(n^2)$	-	-
Lemke	Pivoting	P-matrix	$\mathcal{O}(2^n)$	$\mathcal{O}(n^2)$	-	-
Minimum map Newton	Iterative	No assumptions	$\mathcal{O}(n^3)$	$\mathcal{O}(n^2)$	Quadratic	Yes
Fischer–Newton	Iterative	No assumptions	$\mathcal{O}(n^3)$	$\mathcal{O}(n^2)$	Quadratic	Yes

There are other types of methods that we have not covered in these notes, such as the instance interior point (IP) methods, trust region methods and continuation methods. One can develop any of these using the QP reformulation of the LCP. For LCPs with non-symmetric coefficient matrices, the problems is not quite as straightforward. We leave these type of methods for future work.

Pivoting and PGS methods are common-place today. The Newton methods we have introduced do offer better convergence behavior than PGS-type methods and run faster than pivoting methods. We speculate that the iterative methods for the Newton equation should fit GPU implementation well and thus benefit from hardware that is well suited for parallel matrix-vector products.

Acknowledgements

Thanks to Jernej Barbič for proof-reading and constructive comments. Thanks to Sarah Niebe, Morten Silcowitz and Michael Andersen for working out many details in implementing and testing the numerical solvers. Thanks to Jeff Trinkle and his team for creating a MATLAB framework we could test our numerical solvers against.

References

[AFC+10] Jérémie Allard, François Faure, Hadrien Courtecuisse, Florent Falipou, Christian Duriez, and Paul G. Kry. Volume contact constraints at arbitrary resolution. *ACM Trans. Graph.*, 29:82:1–82:10, July 2010.

[AO11] Iván Alduán and Miguel A. Otaduy. Sph granular flow with friction and cohesion. In *Proc. of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, 2011.

[AP97] M. Anitescu and F. A. Potra. Formulating dynamic multi-rigid-body contact problems with friction as solvable linear complementarity problems. *Nonlinear Dynamics*, 14:231–247, 1997. 10.1023/A:1008292328909.

[AT08] Mihai Anitescu and Alessandro Tasora. An iterative approach for cone complementarity problems for nonsmooth dynamics. *Computational Optimization and Applications*, Nov 2008.

[Bar89] David Baraff. Analytical methods for dynamic simulation of non-penetrating rigid bodies. *SIGGRAPH Comput. Graph.*, 23(3):223–232, 1989.

[Bar93] David Baraff. Issues in computing contact forces for nonpenetrating rigid bodies. *Algorithmica. An International Journal in Computer Science*, 10(2-4):292–352, 1993. Computational robotics: the geometric theory of manipulation, planning, and control.

[Bar94] David Baraff. Fast contact force computation for nonpenetrating rigid bodies. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 23–34, New York, NY, USA, 1994. ACM.

[Bar95] David Baraff. Interactive simulation of solid rigid bodies. *IEEE Comput. Graph. Appl.*, 15(3):63–75, 1995.

[BBB07] Christopher Batty, Florence Bertails, and Robert Bridson. A fast variational framework for accurate solid-fluid coupling. *ACM Trans. Graph.*, 26, July 2007.

[BDCDA11] Florence Bertails-Descoubes, Florent Cadoux, Gilles Daviet, and Vincent Acary. A nonsmooth newton solver for capturing exact coulomb friction in fiber assemblies. *ACM Trans. Graph.*, 30:6:1–6:14, February 2011.

[BETC12] Jan Bender, Kenny Erleben, Jeff Trinkle, and Erwin Coumans. Interactive Simulation of Rigid Body Dynamics in Computer Graphics. In Marie-Paule Cani and Fabio Ganovelli, editors, *EG 2012*

- *State of the Art Reports*, pages 95–134, Cagliari, Sardinia, Italy, 2012. Eurographics Association.

[Bil95] Stephen Clyde Billups. *Algorithms for complementarity problems and generalized equations*. PhD thesis, University of Wisconsin at Madison, Madison, WI, USA, 1995.

[Bri08] Robert Bridson. *Fluid Simulation for Computer Graphics*. A K Peters, 2008.

[CA09] Hadrien Courtecuisse and Jérémie Allard. Parallel Dense Gauss-Seidel Algorithm on Many-Core Processors. In *High Performance Computation Conference (HPCC)*. IEEE CS Press, jun 2009.

[CK00] Bintong Chen and Xiaojun Chenand Christian Kanzow. A penalized fischer-burmeister ncp-function. *Math. Program*, 88:211–216, 2000.

[Cla90] F.H. Clarke. *Optimization and Nonsmooth Analysis*. Society for Industrial Mathematics, 1990.

[CM11] Nuttapong Chentanez and Matthias Müller. A multigrid fluid pressure solver handling separating solid boundary conditions. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '11, pages 83–90, New York, NY, USA, 2011. ACM.

[CPS92] Richard Cottle, Jong-Shi Pang, and Richard E. Stone. *The Linear Complementarity Problem*. Computer Science and Scientific Computing. Academic Press, February 1992.

[DBDB11] Gilles Daviet, Florence Bertails-Descoubes, and Laurence Boissieux. A hybrid iterative solver for robustly capturing coulomb friction in hair dynamics. *ACM Trans. Graph.*, 30(6):139:1–139:12, December 2011.

[DDKA06] Christian Duriez, Frederic Dubois, Abderrahmane Kheddar, and Claude Andriot. Realistic haptic rendering of interacting deformable objects in virtual environments. *IEEE Transactions on Visualization and Computer Graphics*, 12(1):36–47, January 2006.

[DLFK00] Tecla De Luca, Francisco Facchinei, and Christian Kanzow. A theoretical and numerical comparison of some semismooth algorithms for complementarity problems. *Computational Optimization and Applications*, 16:173–205, 2000.

[Ebe10] David H. Eberly. *Game Physics*. Morgan Kaufmann, 2 edition, April 2010.

[Erl07] Kenny Erleben. Velocity-based shock propagation for multibody dynamics animation. *ACM Transactions on Graphics (TOG)*, 26(2):12, 2007.

[Erl11] Kenny Erleben. num4lcp. Published online at code.google.com/p/num4lcp/, October 2011. Open source project for numerical methods for linear complementarity problems in physics-based animation.

[Fis92] A. Fischer. A special newton-type optimization method. *Optimization*, 24(3-4):269–284, 1992.

[FK98] Michael C. Ferris and Christian Kanzow. Complementarity and related problems; a survey. Technical report, University of Wisconsin – Madison, 1998.

[FM99] Michael C. Ferris and Todd S. Munson. Interfaces to path 3.0: Design, implementation and usage. *Comput. Optim. Appl.*, 12:207–227, January 1999.

[GZO10] Jorge Gascón, Javier S. Zurdo, and Miguel A. Otaduy. Constraint-based simulation of adhesive contact. In *Proc. of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, 2010.

[Hec04] Chris Hecker. Lemke's algorithm: The hammer in your math toolbox? Online slides from Game Developer Conference, accessed 2011, 2004.

[Kip07] Peter Kipfer. *LCP Algorithms for Collision Detection Using CUDA*, chapter 33, pages 723–730. Number 3 in GPU Gems. Addison-Wesley, 2007.

[KK98] Christian Kanzow and Helmut Kleinmichel. A new class of semismooth newton-type methods for nonlinear complementarity problems. *Comput. Optim. Appl.*, 11(3):227–251, December 1998.

[KSJP08] Danny M. Kaufman, Shinjiro Sueda, Doug L. James, and Dinesh K. Pai. Staggered projections for frictional contact in multibody systems. *ACM Trans. Graph.*, 27(5), 2008.

[Lac03] Claude Lacoursiere. Splitting methods for dry frictional contact problems in rigid multibody systems: Preliminary performance results. In Mark Ollila, editor, *The Annual SIGRAD Conference*, number 10 in Linkøping Electronic Conference Proceedings, November 2003.

[Löt84] Per Lötstedt. Numerical simulation of time-dependent contact and friction problems in rigid body mechanics. 5(2):370–393, 1984.

[Man84] Jan Mandel. A multilevel iterative method for symmetric, positive definite linear complementarity problems. *Applied Mathematics and Optimization*, 11(1):77–95, February 1984.

[Mur88] Katta G. Murty. *Linear Complementarity, Linear and Nonlinear Programming*. Helderman-Verlag, 1988.

[NW99] Jorge Nocedal and Stephen J. Wright. *Numerical optimization*. Springer Series in Operations Research. Springer-Verlag, New York, 1999.

[OGRG07] Miguel A. Otaduy, Daniel Germann, Stephane Redon, and Markus Gross. Adaptive deformations with fast tight bounds. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*, SCA '07, pages 181–190, Aire-la-Ville, Switzerland, Switzerland, 2007. Eurographics Association.

[OTSG09] Miguel A. Otaduy, Rasmus Tamstorf, Denis Steinemann, and Markus Gross. Implicit contact handling for deformable objects. *Computer Graphics Forum (Proc. of Eurographics)*, 28(2), apr 2009.

[Pan90] Jong-Shi Pang. Newton's method for b-differentiable equations. *Math. Oper. Res.*, 15(2):311–341, 1990.

[QS93] Liqun Qi and Jie Sun. A nonsmooth version of newton's method. *Math. Programming*, 58(3):353–367, 1993.

[rpi] rpi-matlab-simulator. Published online at code.google.com/p/rpi-matlab-simulator/.

[Saa03] Yousef Saad. *Iterative Methods for Sparse Linear Systems, 2nd edition*. SIAM, Philadelpha, PA, 2003.

[Sch94] Stefan Scholtes. Introduction to piecewise differential equations. Prepring No. 53, May 1994.

[SKV+12] Breannan Smith, Danny M. Kaufman, Etienne Vouga, Rasmus Tamstorf, and Eitan Grinspun. Reflections on simultaneous impact. *ACM Trans. Graph.*, 31(4):106:1–106:12, July 2012.

[SNE09] Morten Silcowitz, Sarah Niebe, and Kenny Erleben. Nonsmooth newton method for fischer function reformulation of contact force problems for interactive rigid body simulation. In *Proceedings of Virtual Reality Interaction and Physical Simulation (VRIPHYS)*, November 2009.

[SNE10] Morten Silcowitz, Sarah Niebe, and Kenny Erleben. A nonsmooth nonlinear conjugate gradient method for interactive contact force problems. *The Visual Computer*, 2010.

[ST96] David E. Stewart and Jeff C. Trinkle. An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and coulomb friction. *International Journal of Numerical Methods in Engineering*, 39(15):2673–2691, 1996.

[Sta99] Jos Stam. Stable fluids. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '99, pages 121–128, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[TBV12] Richard Tonge, Feodor Benevolenski, and Andrey Voroshilov. Mass splitting for jitter-free parallel rigid body simulation. *ACM Trans. Graph.*, 31(4):105:1–105:8, July 2012.