Proiect cofinanțat din Fondul Social European prin Programul Operațional Sectorial Dezvoltarea Resurselor Umane 2007-2013
Investește în oameni!
Proiect InnoRESEARCH - POSDRU/159/1.5/S/132395
Burse doctorale și postdoctorale în sprijinul inovării și competitivității în cercetare

UNIVERSITATEA **POLITEHNICA** DIN BUCUREŞTI
Facultatea de Automatică și Calculatoare
Catedra de Calculatoare

Nr. Decizie Senat 243 din 12.05.2016

TEZĂ DE DOCTORAT

Simularea unificată a corpurilor rigide și flexibile folosind dinamica bazată pe constrângeri

Unified Simulation of Rigid and Flexible Bodies Using Constraint Based Dynamics

Autor: ing. Mihai Frâncu
Conducător de doctorat: prof. dr. ing. Florica Moldoveanu

COMISIA DE DOCTORAT

Preşedinte	Prof. dr. ing. Adina FLOREA	de la	Universiteatea POLITEHNICA Bucureşti
Conducător de doctorat	Prof. dr. ing. Florica MOLDOVEANU	de la	Universiteatea POLITEHNICA Bucureşti
Referent	Prof. dr. ing. Vasile MANTA	de la	Universitatea Tehnică „Gheorghe Asachi” din Iași
Referent	Prof. dr. ing. Dan NEGRUȚ	de la	University of Wisconsin-Madison, SUA
Referent	Prof. dr. ing. Ioan SLUȘANSCHI	de la	Universiteatea POLITEHNICA Bucureşti

Bucureşti 2016

Rezumat

Această teză prezintă un cadru unificat atât teoretic cât și practic de simulare a corpurilor rigide și flexibile. El folosește teoria dinamicii bazată pe constrângeri împreună cu progrese mai recente în ecuații diferențiale algebrice și inegalități variaționale diferențiale. Un efort însemnat este pus în exprimarea tuturor modelelor de material ca o singură problemă neliniară de minimizare constrânsă. Sunt derivate rezolvatoare noi, sunt folosite noi metode de integrare precum Newmark și este propusă o nouă metodă de amortizare.

Un alt pilon al cercetării noastre este dinamica bazată pe poziții (PBD). În această teză demonstrăm ca metoda este fizic corectă, este echivalentă cu integrarea Euler implicită și poate fi reformulată ca o minimizare pornind de la principiile variaționale ale mecanicii. Mai mult, când este folosită regularizarea, constrângerile devin mai slabe și complet echivalente cu forțe elastice integrate implicit. Aceasta este și baza pentru noul nostru rezolvator de element finit ce se bazează pe proiecția pozițiilor și este corect din punct de vedere fizic.

În cele din urmă, tratăm subiectul contactului cu frecare în contextul dinamicii non-netede. Arătăm că tratatea contactului de către PBD este de fapt o iterație de punct fix a deja doveditei scheme de pășire în timp la nivel de viteză. De asemenea, demonstrăm convergența acestei iterații și includem și un model mai riguros de frecare. Suntem primii care exprimă PBD ca o problemă de minimizare neliniară și convexă cu constrângeri conice. Această formulare include toate tipurile de constrângeri bilaterale și, mai ales, contactul cu frecare.

Abstract

This thesis presents a unified framework at both theoretical and practical level for simulating rigid and flexible bodies. It uses constrained dynamics theory together with more recent advances in differential algebraic equations and differential variational inequalities. A particular effort is put into expressing all the material models as a single nonlinear constrained minimization problem. New solvers are derived, new integration methods like Newmark are used and a new damping method is proposed.

Another pillar of our research is position based dynamics (PBD). In this thesis we prove that the method is physically correct, it is equivalent to implicit Euler integration and can be recast as a minimization starting from the variational principles of mechanics. Moreover, when regularization is employed the constraints become softer and fully equivalent to implicitly integrated elastic forces. This is also the basis for our novel physically correct finite element solver that relies on position projection.

Finally, we treat the subject of contact with friction in the context of nonsmooth dynamics. We show that PBD contact handling is in fact a nonlinear fixed point iteration of the established velocity time stepping scheme. We also prove convergence of this iteration and include a more rigorous friction model. We are the first to express PBD as a convex nonlinear minimization problem with conic constraints. This formulation encompasses all types of bilateral constraints and, more importantly, nonsmooth frictional contact.

Acknowledgments

I would like to thank my advisor for accepting me for this PhD and giving me the opportunity and freedom to work on the topics I had in target. I am also grateful to my girlfriend Cristina for standing by my side during this low-budget academic break from the industry. Also many thanks to professors Dan Negruț and Mihai Anițescu for their willingness to talk to me and help me effectively. Much appreciation goes to Alin Dumitru and Liviu Dinu from Static VFX for their help on rendering and their nice company. I also had very productive chats with Hammad Mazhar, many of which produced actual results, and I thank him for that. There are also other people that have helped me in some way or another during this thesis, even if with just a piece of advice or a nice conversation, and I hope I can mention them all: Kenny Erleben, Victor Asavei, Anca Morar, Lucian Petrescu, Teodor Cioacă, Horea Cărămizaru, Andrei Craifăleanu, Sergiu Crăițoiu, Vasile Brovcenco, Alessandro Tasora, Dario Mangoni.

The work has been partially funded by the Sectoral Operational Programme Human Resources Development 2007-2013 of the Ministry of European Funds through the Financial Agreement POSDRU/159/1.5/S/132395.

Contents

1 Introduction **1**

1.1 Motivation 2

1.2 Context 4

1.3 Goals 5

1.4 Simulation criteria 7

1.5 Thesis outline 9

1.6 Publications in connection with this thesis 11

2 Related work **13**

2.1 State of the art 13

2.2 Nonsmooth dynamics 18

3 Equations of motion **23**

3.1 Newton equations of motion 23

3.2 Lagrange equations of motion 25

3.3 Rotation kinematics 27

3.4 Newton-Euler equations for rigid bodies 30

3.5 Continuum mechanics 31

4 Time discretization **35**

4.1 Numerical integration 35

4.2 Variational and symplectic integrators 38

4.3 Integration as minimization 39

5 Material models **43**
5.1 Rigid bodies . . . 43
5.2 Elasticity . . . 47
5.3 Threads . . . 48
5.4 Cloth . . . 49
5.5 Virtual try on for clothes . . . 54
5.6 Deformable bodies . . . 56
5.7 Fluids . . . 57
5.8 Granular matter . . . 58
5.9 Collision detection . . . 59

6 Constrained dynamics **65**
6.1 Constraints . . . 65
6.2 Differential algebraic equations . . . 68
6.2.1 Mechanical engineering . . . 69
6.2.2 Molecular dynamics . . . 70
6.2.3 Computer graphics . . . 72
6.3 Velocity time stepping . . . 73
6.4 Nonlinear minimization . . . 75
6.5 Variational minimization structure . . . 78
6.6 Solvers . . . 80
6.6.1 Relaxation . . . 81
6.6.2 Krylov subspace methods . . . 83
6.6.3 Accelerated Jacobi . . . 84
6.7 Applications . . . 89
6.8 Regularization . . . 90
6.9 Energy dissipation and damping . . . 93
6.10 Stability . . . 95
6.11 Constraint based FEM . . . 96
6.12 Unilateral constraints . . . 100

7 Nonsmooth dynamics **103**
7.1 Mathematical prelude . . . 104

7.2 Continuous setting 106
7.3 Polyhedral friction cone 109
7.4 Smooth friction cone 111
7.5 Position projection 113
7.6 Projected iterative solvers 115
7.7 Rigid bodies 116
7.8 Friction models 118

8 Unified simulation framework **121**
8.1 Nonlinear constrained dynamics 121
8.2 Implementation and results 122
8.3 Mixing PBD and VTS 134

9 Conclusions and future work **137**
9.1 Conclusions 137
9.2 Contributions 139
9.3 Future work 140

List of Figures

4.1 The energy evolution over 500 frames of a 15×15 piece of cloth using NCG (green), PBD (purple) and CG (red) and exact (blue) linearly implicit solvers. 42

5.1 Hierarchy of articulated rigid bodies with spherical joints. . . . 45
5.2 Boxes falling on ground solved with position projection. 47
5.3 Mass-spring approximation of a vibrating string. 49
5.4 Links structure of cloth relative to one particle (white one in the center): stretch links (red), shear links (blue), bend links (green). 50
5.5 100×100 piece of cloth hanging from corners and falling over a sphere; simulated in real time at 60 Hz using the conjugate residuals solver. 51
5.6 Simulation of (a) a cloth model consisting of 6910 vertices and 13674 triangles using soft constraints and (b) a garment exported from Marvelous Designer and rendered in OGRE. . . 52
5.7 Two snapshots of a side by side real-time simulation of two 40×40 cloth pieces with the same Young's modulus E: regularized FEM constraints (left) and soft links (right); superimposed in purple is the strain map. FEM offers more realistic folds and the strain is better distributed throughout the cloth. 53
5.8 Flexible cow falling on ground. 57
5.9 Granular matter falling over or trough various meshes. 62
5.10 Simulation of 1000 particles falling on a 20×20 piece of cloth fixed at its corners (using the Sequential Positions method). . 63

5.11 Handling of cloth-mesh collisions and self collision. 63

6.1 Plot of the residual using 3 different solvers. 76

6.2 Plot of β over 20 iterations of the CR solver: original formula clamped below 1 (blue) and power function approximation (red). 87

6.3 Plot of the total stretching at equilibrium of a 50×50 piece of cloth relative to the number of iterations for GS (red) and optimized CR (blue). 88

7.1 Particle contact point with friction cone Υ given by $\theta = \arctan \mu$ and its polar cone Υ° depicted below. 107

8.1 Rigid bunnies falling on a piece of cloth with two way coupling.123

8.2 Flexible dragon falling on stairs and hitting rigid boxes. 124

8.3 Evolution of cloth from initial state to steady state painted as L1 norm of the constraint error for 30 iterations per frame using GS (blue), MINRES (red) and CR (green). 125

8.4 Plot of the simulation cost per frame in milliseconds (vertical axis) relative to the number of iterations (horizontal axis) for GS (red) and CR (blue). 126

8.5 Cloth falling freely over a sphere with friction coefficient $\mu = 0.5$: (a) using accurate friction inside the iterative solver the cloth remains stable on the sphere and (b) using the traditional PBD friction handling (velocity post-processing) method the cloth falls very quickly off the sphere. 127

8.6 Sand piles formed by dropping 3000 particles using VTS with different friction coefficients (15 iterations, Baumgarte stabilization $\gamma = 0.5$). 127

8.7 Plot of constraint error (L1 norm) for PBD cloth simulation with different solvers (frame number on the horizontal axis): Gauss-Seidel (blue), SOR (green) with $\omega = 1.2$, and improved Jacobi (red) with $\omega = 0.5$, $a = 1$ and $b = 0.6$. 128

8.8 Plot of unilateral constraint error (L1 norm) for 3000 particles falling in a box (VTS, $\gamma = 0.5$): Gauss-Seidel (blue), improved Jacobi with $\omega = 0.5$, $a = 1$ and $b = 0.6$ (red), improved Jacobi with $a = 2$ (green), and Sequential Positions using GS (purple).129
8.9 Plot of unilateral constraint error (L1 norm) for 3000 particles in a box with varying masses and friction: GS (blue), improved Jacobi (red) with $\omega = 0.4$, $a = 1$, $b = 0.6$. 130
8.10 Plot of relative velocity along constraints relative to the number of iterations for 1000 particles falling in a box with contact and friction: Gauss-Seidel (blue), improved Jacobi (red) with $\omega = 0.5$, $a = 1$ and $b = 0.6$, and Sequential Positions using GS with $k_c = 1$ and $k_v = 0.1$ (green). 131
8.11 Total energy evolution in time for the simulation of a 10×10 rubber cloth ($\kappa = 2$ N/m, 25 iterations) using NCG implicit integration (blue), regularized PBD (red) and regularized energy preserving projection (green). 132
8.12 Damping response for the simulation of a 40×40 piece of cloth ($\kappa = 2000$ N/m, 25 iterations) using regularized PBD (blue), aggressive damping (red) and slightly damped energy preserving projection (green). 133
8.13 Windows application written in MS Visual C++ using OpenGL.134
8.14 Hybrid method - VTS for contacts after PBD for links 136

List of Algorithms

1 NCG implicit solver 41
2 Nonlinear Minimum Residual 85
3 Nonlinear Conjugate Residuals 86
4 Pseudo-code for computing the internal forces inside a tetrahedron. Here a block Gauss-Seidel approach is employed. . . . 97
5 Pseudo-code for computing the normal and friction forces between 2 rigid bodies in contact. Can be used with either a Jacobi or a Gauss-Seidel approach ($\omega \geq 1, \beta = 0$). 117
6 Nonlinear projected gradient descent constraint solver using a Jacobi approach. 122

List of Tables

8.1 CPU time (for one simulation frame) comparison between Gauss-Seidel and accelerated Jacobi nonlinear constrained dynamics solvers (dual core). 123
8.2 CPU time (for one simulation frame) comparison between Gauss-Seidel and accelerated Jacobi nonlinear constrained dynamics solvers (quad core). 124
8.3 Frame time measurements made on a Intel Core i7 3770 CPU (single-threaded) for the two presented scenarios: hanging cloth (PBD) and falling particles (VTS). 129

Chapter 1

Introduction

This thesis describes and extends a number of methods for mechanical simulation in the context of computer graphics. The theory behind these methods belongs to a relatively narrow subfield of mechanics called *constrained dynamics*, i.e. dynamics with added constraints. In particular, we are focusing mainly on position projection methods modeled as mathematical optimization problems. We will introduce and further detail these notions gradually in the following chapters.

Simulation has become an important part of computer graphics since the 90s. By simulation we mean a type of procedural animation where you set the initial conditions and then the system evolves in a physical way without any user control. Examples include the draping of cloth on an object or a human avatar or rigid bodies colliding into each other. Continuous bodies like elastic objects or fluids have become increasingly used in the past decades in 3D animation films and movie visual effects (VFX). Games have also started to use more and more physics simulation since the 2000s. These trends have driven the adoption of more powerful hardware and also parallelization of algorithms.

One could argue that there is not much research left to do in this field given that most type of scenarios have already been simulated (from realistic clothes to enormous oceans). Also, that the branch of physics governing all these phenomena, mechanics, has been understood for more than three

centuries. However, this is not entirely true. Even when simulation first appeared in computer graphics most of the methods were taken from science and engineering results that dated back to the 50s. But nowadays computer scientists are producing new results and are cross-pollinating the other fields. And even before that, in the second half of the 20th century, researchers discovered with the advent of computers that there are still many things that we do not understand about classical mechanics. The results over the last decades in elasticity and the finite element method, contact dynamics, numerical integration and even chaos theory prove that there is still a fertile ground for research in mechanical simulation. And of course there is the goal of faster algorithms.

Needless to say that we play only a humble role and our contribution only tries to advance a small niche of simulation. We only deal with macroscopic scale simulations and try to rely on constraints rather than forces. If this is a sufficiently accurate view of the world remains to be seen and it is up for debate. It is best to stress from the start that there is no such thing as a perfect simulation and there are competing methods with different pros and cons for all types of applications. We did our best to keep our simulations robust and plausible while at the same time fast and simple to understand, implement and maintain.

In this chapter we will give the motivation behind the thesis, the background and the context of our research and the goals we set at the beginning. Finally we give a quick outline of the thesis and our contributions.

1.1 Motivation

The author's background is in games and physics for 3D games. This is why in this thesis a strong emphasis is laid upon real time application. Also, constrained dynamics is often the method of choice of most rigid body physics engines. This is not always true for deformable bodies where different methods are used. This is what lead us to the idea of unification of methods so that a single engine is used for all simulations and also coupling is achieved between rigid and flexible materials.

One of our initial questions was why are springs utilized for simulating threads and ropes and contact is often handled separately by constraints and complementarity? Why not use constraints also for the rope segments? Or if one uses rigid links, why would she uses penalty methods for contact? And bottom line, why are cloth solvers created as separate products and use almost fully different methods for simulation? This was happening back when constrained dynamics and position based methods were gaining traction in games and computer graphics, but not every one knew about them, how to use them and how they relate to the more established spring-damper methods. In fact this is still true nowadays, though many advances have been made, but a lot of practitioners still consider position based methods (and even velocity based ones) as non-physical.

Another question we asked ourselves was why contact handling techniques based on constraints on nonsmooth dynamics have become so popular in games and visual effects (see Havok, PhysX, ODE, Bullet) and they are still viewed with reluctance by researchers doing elasticity based simulations in both engineering and computer graphics? Is the method not mature enough? Probably so, as these methods only started appearing the late 70s, were developed in the 90s and were implemented after the year 2000. Also, we do not deny the viability of hybrid methods where elasticity is simulated by implicit integration and contacts are handled through constraints. The two different solvers are coupled through co-simulation, a strategy that is often used to accommodate different simulators developed independently under the same roof of multi-physics.

Even today a lot of researchers and commercial products implement their dynamics engines using penalty methods despite their drawbacks. Maybe after all, penalty methods are not that bad and can be made robust as some papers indicate. We did consider that we are biased in our approach given the background described above. But one cannot stop and wonder: why is there such a gap between penalty and Lagrange multiplier methods? What if they are somehow the same? And we think we were able to prove this in our thesis or at least bring more arguments to the hypothesis.

1.2 Context

This work was elaborated in the context of computer graphics with a focus on virtual reality and interactivity. In contrast to engineering we perform no validation, but rely only on visual inspection and plausibility. Still, we did our utmost best to start from the same physical principles and obey the same physical laws as the other older and accepted methods used in engineering and scientific simulations.

Position based dynamics (PBD) became very popular in the last decade. This was one of the few cases where new methods were developed in the games and computer graphics and did not come from a more scientific origin. But it also laocked rigor. We went on a quest of showing what PBD really is and to prove that it is in fact physically correct (for whatever that means). We found out that a lot of people, especially in the engineering community, like to work and explain phenomena with forces, accelerations and constitutive relations like Hooke's law for springs. Everything else, like projection on the constraint manifold, seems artificial, unphysical or cheated. Ideal constraints like non-penetration, constant length or rigidity for that matter are artificial mathematical constructs but they are not far from the truth. We continue the approach in Claude Lacoursière's thesis in that constraints are only as perfect as diamond is, but also that a lot of forces of nature are so stiff that they can be considered as rigid internal forces (action-reaction pairs). Lacoursiére uses this approach to justify the use of regularization for solving the resulting problems. We are turning this argument on its head and argue that regularization is essential for proving the physical roots of PBD. Moreover, even if we do not care and accept ideal constraints as real, we can still show that PBD is a sound method that can be derived from a discretization of the variational principles of mechanics. We took this idea from the same thesis and many other sources, but again used it for a different goal: prove that most simulations can be reduced to a quadratic minimization problem with nonlinear or conic constraints.

Deformable bodies are another thorny subject in computer graphics. A lot of effort is put into methods that look plausible, they are very robust and

employ a lot of tricks rather than trying to be accurate. This is strange because a lot of these methods come from engineering where they have already been validated. So, on the other hand, the tendency nowadays is to go back to these origins and squeeze more accuracy from them even in graphics along with the progress of hardware. But hacks and tricks are still used and this is why there is this distinction between physical and non-physical (or fake) methods. For example PBD methods for soft bodies are considered in the former category, but this is arguable because PBD itself is physical but the constraints themselves can be non-physical. And the example that comes to mind is the modeling of soft bodies as volumetric lattices of springs or links between particles: this is not physically accurate for either the force or the constraint approach. The most accepted method for simulating deformable bodies over the past many decades and in scientific circles is the finite element method (FEM) or finite element analysis (FEA). In order to show the physicality of the position projection method we tried to build a proof in this thesis that accurate FEM simulation is indeed possible using such methods.

Given all these we are ready to define the scope of our research. We are focusing only on constraint based methods in order to improve them. We are also comparing them to more traditional elasticity based approaches but these are only touched marginally. For solving the numerical problem we focus on mathematical optimization methods and in particular on iterative solvers and gradient descent approaches. Going deeper, we concentrate only on the dual formulation of the problem in terms of Lagrange multipliers and matrix-free formulations for quick, parallel and low memory footprint implementations.

1.3 Goals

As already hinted, our main goal was to simulate as many different phenomena possible using the framework of constraints. This thesis was intended as proof of concept and limited itself to coupling rigid and deformable bodies inside the same solver. We hope that it can be extended to other contexts like fluid simulation, grid based continuous materials and other more com-

plex methods used in engineering. Two-way coupling of rigid and deformable bodies is not a new result and we do not claim it. Our contribution is that we use only one numerical framework to model these physical phenomena. Also it may seem that constraint based techniques for simulating rigid and deformable bodies have been around for a while now. We do not deny that, but stress the fact that they have never been unified under the same formulation and proven to be physically correct. Also we developed new methods for rigid bodies, cloth and FEM.

Another challenge we set ourselves was to make these methods suitable for real-time and interactive applications. Our intuition was that if constraint methods proved so successful in games and VFX then they are clearly the way to go and should be extended. We have not yet found an answer to whether these methods are significantly faster than spring-damper or penalty methods. We do think they are more intuitive and easier to implement and rid oneself of many intricacies like boundary conditions.

Not all the simulations presented in this thesis ran fully in real-time, but as a rule we tried not to go below one frame per second and get as close as possible to 60 Hz (which is the de facto simulation rate for smooth animation in games[1]). This is why most of our integrators work with a time step of 16 ms. Our aim was robustness and stability at such large time steps. Then optimizing the algorithms for speed would be just another step, which we have taken in many cases.

The biggest condition for reaching interactive frame rates is that our algorithms are parallel and scale well with the number of computing units. A lot of the established methods in simulation are sequential and relied on Moore's law to get faster every year. Simulating a scene in real time also depends on the number of objects or their complexity. This is why parallel computing is the way of the future: add more processing power for more bodies while keeping the simulation time constant. Old sequential programming tricks are no longer sufficient to boost simulation speed. Many times they are not even suitable for parallel implementation because of the causal links between data or code. This is why we set ourselves the goal to develop

[1]Probably even faster rates will be needed for VR headsets.

new and competitive solvers that work well on multi-core and many-core (i.e. GPGPU) architectures directly from the mathematical level.

1.4 Simulation criteria

In order to decide which of the existing or possible methods are most suited for simulating bodies in contact we need to clarify some criteria. Of course, computation speed has always been a criterion and it usually induces a trade off with accuracy. So there is no clear way of choosing the best method: we always have to make compromises. So we end most of the times up doing a multi-criterion analysis hoping to find a set of Pareto optimal methods.

Let us enumerate some of these criteria:

- *stability*: above all we want the simulation to not blow up, and then we desire to obtain rest configurations that do not jitter. Unfortunately this word has a wide encompassing meaning as it is connected to many causes, e.g. dissipativity, integration methods, solver convergence etc.

- *accuracy*: this is usually sought in mechanical engineering and often discarded in computer graphics. The truth lies in between, as there is no such thing as total accuracy in numerical modeling, but better accuracy many times means more correct simulation of natural phenomena - which the eye can discern. The problem here again lies in the fact that there are several different quantities that need to be accurate at the same time, e.g. non-penetration constraint, friction, momentum conservation etc.

- *conservation laws*: we refer here to those quantities that are invariants of motion, sometimes called first integrals and arising from symmetries. The usual examples are energy, linear and angular momentum and spatial symmetry of the motion. This of course becomes blurry when dissipation comes into play - and it usually does due to numerical schemes and is usually desired for stability reasons. One would expect that a method that handles well the elastic case should do just as well

for the inelastic one. But much greater effort is needed to achieve this kind of "structure preserving" methods.

- *performance*: surely we would like to achieve the smallest computational cost without sacrificing too much of the above criteria. How much is too much varies from field to field. But it is clear that long simulation times usually translate in the end to money being spent and one always prefers a smaller cost for the same outcome. Performance becomes a hard constraint in the case of real time applications - this is often mitigated by the number of objects in the scene. Typical interactive rates are above 10 Hz (preferably 30 or 60 Hz), meaning that the whole simulation frame should take place in milliseconds. And given the power wall in front of Moore's Law the only viable approach nowadays is through parallel computing.

- *robustness*: this can be viewed as a subset of stability, but we wish to give it a more precise and separate meaning, i.e. the ability to recover from ill conditioned configurations in an elegant, silent and plausible manner.

- *veracity*: this too can be synonymous to accuracy, but again we mean something apart from that, namely physicality and plausibility. It is said of many methods they are not physical, i.e. they are not rooted in physical laws. This is often a subjective argument, it is often ambiguous or false, and is usually due to lack of theoretical or experimental proof. Numerical simulations are notoriously hard to show that they mimic reality accurately or that they adhere to first principles rigorously. This is why plausibility was coined: a simulated motion is valid if it looks good at visual inspection. This criterion has been often overused in computer graphics (the author included), but this does not mean that it is a weak criterion. It still needs to be upheld and used as a fail safe when physical laws are vague or paradoxical (e.g. impact, friction).

- *collision detection*: the quality of this process determines a lot of the behavior of the numerical solver. This is why it is very important to

determine from the start the collision requirements and issues for each particular method, as they not only affect performance, but also all the other criteria listed above.

1.5 Thesis outline

Chapters 3, 4 and 5 are meant as introductory material so that the reader is familiarized with some of the theoretical notions and common practices used in physical simulation before going into the heart of the matter. Chapters 6 and 7 are the crux of our research and represent a mixture between presenting existing methods and our own contributions. The rest of the thesis presents implementation details, results and conclusions with the focus on our unified simulation framework. Contributions are highlighted in the sections where they appear and also at the very end together with references to the articles where they were published.

- **Chapter 2** does a quick survey of the state of the art and also mentions some historical details.
- **Chapter 3** gives a brief overview of the equations of motion for particles, rigid bodies and continua. Some focus is put on Lagrangian mechanics and the kinematics of rotation. The equations presented here are the continuous forms which are later discretized.
- **Chapter 4** treats the subject of numerical integration and gives short descriptions of the types of integrators. We then focus on the implicit Euler integrator and present its minimization form together with our new Nonlinear Conjugate Gradient solver.
- **Chapter 5** is intended to be an introduction to the practices of modeling different types of objects and materials, so that we can focus solely on the mathematics and numerical methods in the following chapters. Even though the stress is on mass-spring systems and finite elements, we also give a survey of other existing modeling and simulation techniques. Additionally, this being a chapter about spatial and geometric

discretization, we also include a section on collision detection techniques. Spread around the chapter are pictures of some of our simulations.

- **Chapter 6** introduces the ideas of constrained dynamics with an emphasis on equality only and holonomic constraints, namely on differential algebraic equations. We chose this separation from contact problems because notions are easier to explain and the resulting problems are usually convex and pose less challenges. This chapter also allows us to treat the simulation of articulated and deformable objects without worrying about contact (which we introduce only at the end). Our contributions include a new parallel solver, proven equivalence between elasticity and regularized position projection and an accurate position based finite element solver.

- **Chapter 7** gives a brief overview of nonsmooth dynamics, namely the necessity of using impulses to handle impacts and friction. Then we proceed to presenting our new nonlinear and fully implicit approach to nonsmooth dynamics yielding a rigorous formal model of position projection with frictional contact. Our main contribution in this chapter is a position-based rigid body simulator with mathematically correct friction handling.

- **Chapter 8** describes the actual theoretical foundations and practical details to build our unified simulator. Basically it puts together everything from the previous chapters. We have included some visual results, some measurements and some performance timings. We also added some numerical experiments we performed during our research to analyze some issues in isolation. We end with a discussion about our experiences and failures in mixing velocity and position based methods.

- **Chapter 9** gives the final conclusions and summarizes our work. We enumerate once more our contributions and finally list quite a few issues that are still outstanding and represent challenges for the future.

1.6 Publications in connection with this thesis

A good part of the research presented in this thesis was also published in the following articles:

- *Minimum residual methods for cloth simulation* [FM14b]
- *An Improved Jacobi Solver for Particle Simulation* [FM14a]
- *Cloth Simulation Using Soft Constraints* [FM15a]
- *Virtual Try On Systems for Clothes: Issues and Solutions* [FM15b]
- *Simulating Large Scale Coupled Granular Material Simulations using Position Based Dynamics* [MFN16]

Chapter 2

Related work

Now that we clearly defined the context of our thesis we can restrict ourselves to reviewing only a subset of the existing dynamics simulation literature. This chapter will only cover very high level aspects of the state of the art in the topics connected to our research. More details will be given throughout the thesis which can be considered at times as a state of the art review.

2.1 State of the art

Rigid bodies with contact can be simulated with constrained dynamics and other methods, most importantly the penalty method [BZX14]. Other approaches consider the rigid body as a collection of particles and contact forces are computed using the discrete element method (DEM) or in other ways [TSIHK06, Jak01]. The particles can move under rigid transformations [Har07] or be constrained together to form a composite rigid object [Cou12, MMCK14].

There has been a wealth of work published on the subject of rigid body simulation with contact and friction - for a survey see [BETC14]. We note the advances made in the 90s by Baraff, Stewart and Anițescu (to name a few). Given the drawbacks of penalty forces, Baraff introduced the acceleration based *linear complementarity problem* (LCP) method (inspired by earlier work from Lötstedt) [Bar94]. This method had its problems too (related

to impacts and the Painlevé paradox) that were later solved by a velocity based approach that allows discontinuities in the velocities, i.e. impulses. This approach was developed in the 80s by Moreau [Mor88] under the name *nonsmooth dynamics* and was later extended by others [Bro96, AB08, Stu09]. The new *velocity time stepping* (VTS) schemes [ST96, AP97, AH04a] became very popular in computer graphics, games and real time simulators [TBV12, Erl07, Cat05]. We take a similar approach in this thesis, but based on more recent work geared towards convex optimization [TA11, MHNT15]. Other methods that employ both optimization and a fixed point iteration can be found in [AH04b, ACLM11] or the slightly different Staggered Projections approach in [KSJP08].

The underlying mathematical problem has many names. In game physics it is often referred to as LCP, but this name has become slightly obsolete as not every method is LCP based anymore. In fact, many formulations use a type or another of *nonlinear complementary problem* (NCP) [ST96, SHNE10a] or equivalently a *variational inequality* (VI) [CPS92]. The continuous problem is actually called a *differential complementarity problem* (DCP) or *measure differential inclusion* (MDI) or *differential variational inequality* (DVI) depending on the authors. The newer convexified approach can be expressed (in the discrete case) as a *quadratic program* (QP) with conic constraints or as a *cone complementarity problem* (CCP).

Granular matter has been an area of research in computational mechanics for decades. The method of choice is usually the *discrete element method* (DEM) which treats the granules as elastic billiard balls and uses Hertzian contact theory [GS02]. The DEM method was used in graphics too [BYM05, ATO09, Har07]. Another approach was a continuum based one, considering the granular matter a special kind of fluid [ZB05, NGL10]. This was followed by a Lagrangian version derived from the *smooth particle hydrodynamics* (SPH) method for simulating fluids [AO11, IWT12].

An alternative to DEM is the nonsmooth dynamics approach where the particles are considered fully rigid and this is the path we are following. In fact the method was developed for the more general case of rigid bodies, but that can be turned into an advantage given the granules can have any shape

other than spherical [BYM05]. A great deal of articles have been written on the subject of multibody dynamics with contact and friction and also explicitly on the subject of granular flow [TA10, RA05, LSB10].

Deformable bodies have been traditionally simulated using implicit integrators due to their unconditional stability properties. These have been applied not only to mass-spring systems, but also to simulations using the *finite element method* (FEM) [MSJT08]. Recently, the popular Backward Euler integrator has been recast as an optimization problem [BML+14] helping us to gain new insights on a problem that used be solved solely by the Newton method. A new alternative that is totally different from implicit integration of elastic systems is *exponential integration* [MSW14] which relies on evaluating trigonometric matrices (in terms of exponential functions).

Cloth is one example of a deformable body modeled as a mass-spring system [Pro96]. The implicit integration of the equations of motion has become pervasive for cloth since the seminal work of [BW98]. Its main attraction is its unconditional stability for very stiff equations and large time steps. By implicit integration we usually mean the implicit Euler method, but other implicit integrators were also employed, like BDF-2 [CK02], Implicit Midpoint [OAW06] or Newmark [SSB13]. These integration methods offer better energy conservation and more responsive simulation, in contrast to implicit Euler which artificially dampens out high frequency details in exchange for stability. Other variations include approximations made to the force Jacobian [HE01] or an implicit-explicit (IMEX) approach [EEH00, BMF03]. Most approaches however use only one Newton solver iteration. In games *position based dynamics* (PBD) is usually preferred for simulating cloth [Jak01, MHHR07, GHF+07]. The approach we took is in the same vein as [BBD09] and [How11] but we do only one pass for both velocity and position. Soft bodies can also be modeled by adding internal pressure to a closed cloth mesh.

Constrained dynamics was not initially considered for simulating deformable bodies, but this changed with the advent of PBD and constraint regularization [SLM06]. PBD was originally introduced by Jakobsen for games based on molecular dynamics methods and a nonlinear version of the Stewart-

Trinkle solver for rigid bodies [Jak01]. Goldenthal later showed that position projection is equivalent to the fully implicit integration of a constrained system [Gol10].

Constraint based methods appeared originally in their acceleration based formulation for rigid body dynamics with joints and contacts [Bar94, BCP96]. Later on, velocity or impulse based methods gained more popularity [AH04a, Erl07]. Position based methods are actually a nonlinear version of velocity based ones, in the sense that they can still be expressed as velocity filters, but constraints are enforced at positional level [ST96]. Part of the inspiration for PBD came from molecular dynamics where methods like SHAKE or RATTLE are widely used [BKLS95]. A more detailed study for the application to cloth simulation in computer graphics was done in [Gol10]. Here the method of *fast projection* is developed based on an implicit treatment of constraint directions [HCJ+05] and a better energy preserving integrator is also derived. Position based methods rely on *projection* for solving differential algebraic equations (DAE), which is ultimately an optimization problem [HLW06]. Another part of inspiration came from *strain limiting* techniques used in elastic cloth simulation [Pro96, BFA02].

Constraint based methods are often criticized for the fact that they simulate only nearly inextensible materials and are prone to *locking*. In order to address this English and Bridson [EB08] use fast projection in conjunction with a BDF-2 integrator on a conforming triangular mesh. They also give a brief proof for fast projection being the limit of infinitely stiff elastic forces. Other authors prefer to use quad-predominant meshes or diamond subdivision [Gol10].

Constraint *regularization* was employed mainly in [Lac07] for making rigid dynamics with contact and friction more tractable numerically. We take the name *soft constraints* from [Cat10] where an older idea is used: regularization under the mask of Constraint Force Mixing (CFM) [Smi06]. Recently constraint regularization has been used for particle based fluid simulation [MM13]. Another application was intended for the simulation of deformable elastic models using a constraint based FEM formulation [SLM06]. Similar position based approaches can be found in [BML+14] and [BKCW14].

The FEM constraint approach is similar in philosophy with *continuum strain limiting* [TPS09, MCKM14].

The idea of a unified solver is not new and our simulator bears maybe most similarity to Autodesk Maya's Nucleus [Sta09]. Our results are also along the line of more recent PBD work [MMCK14, BKCW14, DCB14] and Projective Dynamics [BML+14]. In addition, a great job of emphasizing the role of nonlinearity for achieving stability was done in [KTS+14] and [TNGF15].

Variational integrators are a special class of integrators that can be deduced directly from the discretization of Hamilton's principle and the Euler-Lagrange equations of motion [SD06]. They are also symplectic integrators, i.e. they preserve area in phase space, which also means they are closer to preserving energy and momenta [HLW06]. Many of them are explicit methods (e.g. Symplectic Euler, Verlet, Leapfrog) so care must be taken when choosing the time step size. Variational implicit methods like Implicit Midpoint or Newmark are more stable and can be converted to projection schemes. This is why we used them as inspiration for our energy conservation strategy.

Iterative methods are currently the preferred way of solving constrained mechanical systems for real-time. Using exact methods can become infeasible when adding contact and friction for more than a few hundred bodies [BETC14]. The fastest and most robust iterative method used in the present is Gauss-Seidel (GS) [Cat05, Erl07]. GS also knows improvements such as line search with conjugate directions [SHNE10a] or subspace minimization [SHNE10b]. Jacobi is another relaxation method, closely resembling GS, but it converges slower and needs modifications to remain stable. Still it is preferred to GS for parallel implementations as it can process each constraint independently from the others [TBV12].

The Conjugate Gradient (CG) method has a good reputation for solving linear systems as it has better convergence than matrix splitting methods like Jacobi or GS [Saa03]. Even though it was used for implicit integration of mass-spring models [BW98] it has never gained traction in constrained dynamics simulations. There have been attempts at using it [RA05], but many argued against its applicability for different reasons [Erl07, Ton12,

Mor05]. Our approach is based on a minimum residual variant of gradient descent algorithms as it guarantees decreasing residual energy and is more stable. After optimizing the conjugate residuals algorithm we arrived at a version of Jacobi with improved convergence. A minimum residual method (GPMINRES) was also used in [HATN12]. The line search Jacobi algorithm offers similar improvements to ours [TBV12, CPS92], but our addition of a momentum term bears more resemblance to Nesterov's method [MHNT15].

2.2 Nonsmooth dynamics

We will now give some more attention to the topics of rigid bodies, contact, friction and nonsmooth dynamics, as they represent a big part of the foundation of this thesis. We will also give some historical details. Before starting we would like to point the reader to two bird eye view works on these subjects in the context of computer graphics: the first one is the SIGGRAPH tutorial given by Baraff and Witkin [Bar97, Wit97] more than a decade ago and focused more on acceleration based methods and the other one is a more recent state of the art review [BETC14].

The formulation of contact dates back to Signorini in the context of elasticity. The solution to the problem of contacting elastic bodies was given in the early 60s and contact complementarity conditions were named Signorini-Fichera (or Moreau-Signorini) [WL06]. The application of this conditions to particles and rigid bodies was done in the 70s and 80s mainly by Moreau and Monteiro-Marques [Mor88]. Note that in this thesis we are not tackling the full formulation of elastic contact, but relying on approximations to handle contacts for deformable bodies (just as we did for PDEs). The work of Moreau was later continued by the likes of Jean, Jourdan, Alart, Curnier and others.

In the early 90 Baraff took inspiration from earlier work on complementarity by Lötstedt and formulated one of the first working acceleration based simulators with impacts and resting contact. He modeled frictional contact as a linear complementarity problem (LCP) and used direct solvers like the ones devised by Lemke or Dantzig [CPS92]. His seminal work happened orig-

inally in the context of robotics but it quickly migrated to computer graphics [Bar94]. The problem with this approach was that it only dealt with resting contact, the integrator had to be restarted after impacts and the LCP did not always have a solution. Also, the exact LCP solvers did not scale well over a few hundred objects (the algorithm complexity being exponential).

These problems were later solved by Stewart and Trinkle who based their results on the nonsmooth dynamics developed earlier by Moreau and others. The essence of nonsmooth dynamics is that it can handle discontinuities in the velocities by using advanced mathematical topics like convex and nonsmooth analysis, Lebesgue integration, theory of distributions, vector measures and others. New special contact integrators were developed in this context to handle paradoxes of Painlevé and Zeno and the non-existence of force level solutions, e.g. Moreau's *sweeping process*. These came later to be called *velocity time stepping* (VTS) methods. Such a method was introduced in the now classic paper of Stewart and Trinkle [ST96] by means of semi-implicit integration and a polyhedral friction cone LCP approximation. Although Stewart later gave extensive and intricate proofs to his method, the underlying complex tools of nonsmooth dynamics are not that necessary to understand and implement his numerical scheme.

One could argue that the original Stewart-Trinkle scheme was formulated at position level. Even though initially the positional non-penetration constraint were linearized, a nonlinear fixed point iteration is presented in a later section. This nonlinear solution is one of the first of its kind and it was later referenced by Jakobsen as a setting stone for position based dynamics. The positional formulation was later abandoned in favor of a velocity level complementarity condition from Anițescu and Potra [AP97]. More than that, Stewart later gave a critique of the nonlinear approach saying that it results in random coefficients of restitution [Ste00]. We understand this concern but argue that it also plagues stabilized VTS methods as well as position based ones. We accept this fact just as other authors do and consider the exact coefficient of restitutions in nonsmooth dynamics as an ongoing topic of research [SKV$^+$12].

The Stewart-Trinkle and Anițescu-Potra schemes marked a landmark in

rigid body simulation and stood as inspiration for the physics engines of the 2000s (e.g. ODE, Vortex, Havok, PhysX) that became very popular in games. An important extension to these schemes was added later by Anițescu and Hart in terms of a stabilization term directly inside the LCP formulation [AH04a] in contrast to post-stabilization in a separate LCP step at the end [CP03]. They are not the only who have linearized the constraints in a similar manner: Faure [Fau99], Sauer and Schömer [SS98] and others. This type of stabilization became actually the de facto velocity time stepping scheme for solving contact. Our contribution in this chapter is to show how a nolinear fixed point iteration of this scheme is actually equivalent to position projection and implicitly PBD.

From the late 90s on contributions in the field of complementarity based contact and nonsmooth dynamics happened in two big directions: mechanical engineering and computer graphics (including video games and interactive applications). In this thesis we focus on the latter, but we do have to acknowledge the influential work of mechanical engineers like Tasora, Glocker, Pfeiffer, Klarbring and many others. Good overviews of nonsmooth contact dynamics are given by Studer [Stu09] and Preclik [Pre08]. We also have to mention the comprehensive books of Brogliato and Acary [AB08, Bro96]. Acceleration level formulation persisted though in engineering articles (and not only [RKC02]) and scientists are still slow to adhere to the velocity time stepping formulations for various reasons.

The original discrete formulation of frictional contact involves a nonlinear smooth friction cone (i.e. ice cream cone) and there are numerical schemes that can handle this directly [Jea99, JAJ98]. The LCP polyhedral friction cone formulation on the other hand has a more solid mathematical foundation. The downside of this formulation is that it makes friction anisotropic. The less faces you use the more pronounced this is. And this is particularly the case for real-time implementations. Also, the mixed LCP formulation cannot be converted to a convex optimization. The transformation of the velocity time stepping scheme to a quadratic program with conic constraints can only be done via a convexification process [Ani06, ACLM11, DSF98]. We chose to use the approach of Anițescu and Tasora [TA11] in order to

use isotropic friction in our simulations (despite its potential artifacts). The nonconvexity of the original problem is exemplified by Anițescu in [AH04b]. Kaufman argues that frictional contact can be modeled as two coupled optimization problems (one for normal contact and one for maximum dissipation) that are equivalent to one single nonconvex problem; his approach is to solve these two optimizations in a staggered approach [KSJP08].

Lacoursière [Lac07] and Kaufman [KP12] also tried to derive nonsmooth dynamics from the more general discrete mechanics, i.e. a direct discretization of the variational principle of mechanics. They relied on earlier work from Fetecău [FMOW03], Pandolfi [PKMO02] and Kane [KROM99]. These are all extensions of the variational integration framework that we presented in the previous chapters to dissipative forces and nonsmooth phenomena like contact and friction. We are not dealing with all the complexities of these methods in this thesis, but we use them as inspiration and argumentation towards expressing our unified simulation framework as one big minimization problem with constraints.

In this thesis we are following mostly the work of Anițescu and his collaborators and also variants of it that made it into computer graphics and games. There are not so many publications in this direction [Cat05, Erl07, TBV12] but a lot of knowledge has coalesced by means of open source projects (e.g. ODE, Bullet), presentations at GDC and elsewhere, books, magazine articles, web sites and blog pages. Game physics is also subject to a lot of simplifications and optimizations so care must be taken when it comes to accuracy; still, that does not mean that the underlying mathematics is not correct.

Chapter 3

Equations of motion

We start by setting up some mechanical notions, give some insights for geometrical and physical meanings of the concepts introduced and proceed to presenting the equations of motion in the Newton and Lagrange formulations. We introduce Hamilton's principle and set it as a foundation of later results in this thesis - mainly the idea of looking for an extremum. Rotation kinematics and the Newton-Euler equations of the dynamics of the rigid body are briefly presented and in the end we give a quick guide to continuum mechanics.

3.1 Newton equations of motion

We are not going to introduce all the notation used in this thesis in this section. We are focusing only on the idea of a point mass particle and build from there, just as in Newtonian mechanics [GPS02].

Before mechanics came geometry as a physical science, so it is natural to measure the movement of a particle as the variation of its *position*. Coordinates came much later with Descartes but they became the standard way to describe the position of a point mass: $\vec{x} \in \mathbb{R}^3$. The shape drawn by this point at different moments in time is called a *trajectory* and can always be defined as a parametric curve: $\vec{x}(t), t \in \mathbb{R}$. Just like in differential geometry this curve is described at every point by its tangent vector - if you

know this derivative function you can reconstruct the trajectory by means of integration. This vector gives us two physical meanings: the direction of the movement and its speed at every instant. We call it the *velocity* vector $\vec{v} \in \mathbb{R}^3$. Vector spaces are the natural representation of such physical quantities like position and velocity as the entities remain essentially the same under rotation. The equation $\vec{v} = \dot{\vec{x}}$ is called the kinematics of the particle. It looks innocuous and a mere substitution but it is an important one and things can get more complicated when changing coordinates or moving to rigid bodies with rotations.

The trouble is that we need two quantities to completely specify the state of a particle at any instant. Aristotle thought that position is enough and that interaction is given by velocities but he was proven wrong in the end [Dug57]. Galileo discovered the law of inertia that states that a particle with no external interactions moves with constant velocity, so it means that velocity must change too during more complex motion. This is why we need the pair of position and velocity to describe each motion state. Also, trajectory is deterministic so any state at any moment could be the initial one. Actually the differential problem of dynamics is often called an *initial value problem*. This is where Newton comes in and introduces acceleration as the derivative of velocity: $\vec{a} = \dot{\vec{v}}$ and assigns the force as the agent of interaction, all under the umbrella of differential equations.

But before that, mass is another important concept: it measures inertia and interaction between particles. Mass times velocity gives *momentum*[1]: $\vec{p} = m\vec{v}$, or the quantity of motion - that gets conserved under no external forces. Newton's form of equations of motion give us a law of the variation of momentum when external forces are present. If considering the mass constant and using acceleration we get a second order *ordinary differential equation* (ODE):

$$\dot{\vec{p}} = m\ddot{\vec{x}} = \vec{f}(\vec{x}, \vec{v}, t). \tag{3.1}$$

It is very important to note that the force $\vec{f}$ can depend on the position

[1]This quantity was called by earlier scholars *impetus* [Dug57] and identified as the property that stays with a projectile and continues its motion, acting as a cause or agent of movement.

and/or the velocity which vary in time or even explicitly on time; it can even be constant. So the difficulty arises in the general case when we need to integrate any nonlinear function - this is why we need the numerical integrators introduced in the next chapter. It is easier to do this if converting to a first order system of ODEs. Before doing that we would like to change the notation so that it encompasses all bodies in the system, so we denote by lower case bold letters all vectors in $\mathbb{R}^{dN}$, where N is the number bodies and d is the space dimension, usually 3 but didactic cases also include 2 and 1. In the end the first order equations that we will employ throughout this thesis are:

$$\frac{d}{dt}\begin{pmatrix}\mathbf{x}\\ \mathbf{M}\mathbf{v}\end{pmatrix} = \begin{pmatrix}\mathbf{v}\\ \mathbf{f}(\mathbf{x},\mathbf{v},t)\end{pmatrix}, \tag{3.2}$$

with $\mathbf{x}(t_0) = \mathbf{x}_0$ and $\mathbf{v}(t_0) = \mathbf{v}_0$ as initial conditions. This equation can be even further condensed to $\dot{\mathbf{z}} = \varphi(\mathbf{z}, t)$ with $\mathbf{z}(t_0) = \mathbf{z}_0$ in order to better see the connection with the theory of ODEs and their numerical integration [Str07, PTVF07, HNW87].

3.2 Lagrange equations of motion

The equations in the previous section are expressed in Cartesian coordinates that describe the Euclidean space. They are often called *maximal coordinates*, as bodies are considered free and then the forces between them are calculated. Often, such forces come in pairs and do no work, keeping the bodies in equilibrium, and thus they are called *internal forces*. These forces basically impose constraints on the system and remove some of the degrees of freedom, e.g. a box cannot fall through the ground.

The alternative procedure of Lagrange implies identifying from the start the degrees of freedom of the system and is often called a *reduced coordinates* formulation. Although reduced coordinates formulations can be obtained without knowledge of the Euler-Lagrange equations (e.g. for kinematic chains in robotics [Fea14]), we will deal here with the general framework of Lagrangian dynamics. This is built upon the notion of energy that originated from Leibniz rather than Newton and axiomatic principles of mechanics due

to Maupertuis, d'Alembert, Gauss, Hamilton and others [Lan70].

These reduced coordinates are also known as *generalized coordinates* $\mathbf{q} \in \mathbb{R}^n$, where n is the number of degrees of freedom. They coincide with the aggregated coordinates $\mathbf{x}$ from the previous section when motion is unconstrained and Cartesian coordinates are used. *Generalized velocities* are simply denoted as $\dot{\mathbf{q}}$. From a geometric point of view generalized coordinates form the configuration space and their corresponding derivatives lie in the tangent spaces to this manifold. The other mechanical vector quantities like momentum and force have their generalized counterparts too.

Kinetic energy of a particle is $T = mv^2/2$ and we can also define a potential energy V such that the force acting on the particle is a gradient of this potential, i.e. $\vec{f} = -\nabla V$. Such forces are called conservative as their work does not depend on the path taken (and they also conserve total energy $E = T + V$). Many of the forces in mechanics (in fact all of them at the fundamental level) are conservative, so most of the theory is built upon this assumption[2]. Still, in Lagrangian dynamics there are always ways of adding back dissipative forces. But for now let us limit ourselves to introducing the Lagrangian function as the difference between the kinetic and the potential energy[3]: $\mathcal{L} = T - V$.

For a system of particles (and even in general) the kinetic energy has the form $T = \frac{1}{2}\mathbf{v}^T\mathbf{M}\mathbf{v}$, where $\mathbf{M}$ is the positive-definite mass matrix[4], and the potential energy is a function only on positions $V(\mathbf{x})$. These are more or less the same in generalized coordinates. Then the Euler-Lagrange equations of motion are:

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\mathbf{q}}}\right) = \frac{\partial \mathcal{L}}{\partial \mathbf{q}}. \tag{3.3}$$

There are different ways of obtaining these equations [GPS02, Lan70] but in essence they are not far from Newton's equations: we have a force term on the right hand side and the time derivative of the generalized momentum on the left. The real power of these equations comes from the fact that they

[2]We refer here mainly to Hamiltonian dynamics.

[3]One can work his way backward starting from the properties of the Lagrangian in order to build it [LL60].

[4]It is also diagonal for particles, and predominantly diagonal in the rest of the cases.

are equivalent to a more general extremum principle known as Hamilton's principle[5]. This states that for a finite motion over an interval in time a certain integral quantity has to be stationary (either maximal or minimal) among all other virtual trajectories possible; this quantity is called *action* and it is the integral of the Lagrangian. The principle is written as:

$$\text{extremize } S = \int_{t_1}^{t_2} \mathcal{L}(\mathbf{q}, \dot{\mathbf{q}}, t) dt. \tag{3.4}$$

Another way to view this is similar to calculus where a functions reaches a stationary point when the derivative is zero. As the action is a functional (i.e. integral of a function) a different field of mathematics evolved called *calculus of variations* [Lan70] that gives us a similar condition: $\delta S = 0$. This is read "the variation of S is zero" and it means the value of S is stationary on a trajectory $\mathbf{q}(t)$ given infinitesimal variations of the coordinates $\delta\mathbf{q}$ around it, also known as *virtual displacements*. More details can be found in text books [GPS02]. The importance of Hamilton's principle and its associated notions may appear hard to grasp and many times unnecessary in the context of dynamics simulation. But we will show in the next chapters how it makes a powerful link between the foundations of mechanics and numerical methods employed, especially the ones based on optimization.

3.3 Rotation kinematics

In order to be able to present the equations of motion corresponding to rigid bodies we need to study in more depth rotations or orientations. The words have basically the same mathematically meaning, although they may refer to different things, i.e. a transformation or a property of the reference frame. This is because rotations can be seen as either a change of the position of a point in space or as a conversion of coordinates to a rotated frame of reference.

Surprisingly, rotations bring most of the complexity to the dynamics of

[5]Also popular as the *principle of minimum action* (which is only partially true).

rigid bodies and give birth to intricate mathematical constructs. There is a broad literature on the representation of rotation (also called *attitude* in some fields like aeronautics) [GPS02, Fin09, Mar03]. The most common representation comes from linear algebra and is a 3 by 3 orthonormal matrix: $\mathbf{R} \in \mathbb{R}^{3\times3}, \mathbf{R}^T\mathbf{R} = \mathbf{1}_3$. Maybe the most important property of this matrix is that it has only one real eigenvalue and its corresponding eigenvector is the axis of rotation. This is basically *Euler's theorem of rotation* which states that any rotation can be represented as an angle and an axis of rotation. Also, the determinant of $\mathbf{R}$ must be 1, otherwise we obtain what are called *improper rotations*, i.e. transformations that also include reflections (that change the handedness of the system). A useful note is that the columns of $\mathbf{R}$ represent the axes (unit vectors or versors) of the reference frame; they can also be called *direction cosines*. Rotating a point or transforming it to a different frame of coordinates is done by multiplying a vector by the rotation matrix: $\vec{x}' = \mathbf{R}\vec{x}$. This is the same as doing a dot product with every column vector[6].

The kinematic equation in the rotation matrix representation is:

$$\dot{\mathbf{R}} = \vec{\omega}^{\times}\mathbf{R}, \tag{3.5}$$

where $\vec{\omega}$ is the angular velocity (giving the *instantaneous axis of rotation* and angular speed) and the $^{\times}$ symbol means the skew-symmetric matrix associated with a vector [Bar97]. The product is basically the cross product of $\vec{\omega}$ with every column of $\mathbf{R}$. Generally, the angular velocity cross product has the meaning of a rotational time derivative. We will not deal with details of deriving the concept of angular velocity here, as they are found in many textbooks [GPS02, Fin09]. We want to note though that it is strongly connected to the concept of *infinitesimal rotation* which is in turn an element of a Lie algebra and the finite rotations are elements of a Lie group called the *special orthogonal group*, i.e. $\mathbf{R} \in SO(3)$. These are subtle observations that may not be needed in practice, but knowing them helps when working with rotations.

[6]We assume basic vector operations like dot and cross product are familiar.

Euler also showed that 3 is the minimum number of parameters that describe a rotation - the Euler angles. These are the rotational degrees of freedom. Surely 9 parameters in a rotation matrix are a large number of parameters and they are redundant ones - the orthonormality constraints must be maintained. Ideally, one would use only the 3 unconstrained Euler angles in equations (kinematic and dynamics) and this is often the case in mechanics courses. The downside of Euler angles is that they manifest singularities and gimbal lock [Han06]. There are also 12 different conventions of Euler angles depending on the order of the rotations around which axes. There exist other 3 valued vector representations of rotation: e.g. the Gibbs or Rodrigues vector [Mar03].

Sets of 4 valued parameters have been devised to counter these singularities[7]. Examples include Cayley-Klein parameters [GPS02] and quaternions: $\xi \in \mathbb{H}$. The latter are more relevant to us as they are used widely in dynamics simulation and we will be using them too throughout the thesis. The constraint associated with the quaternions is that they remain of unit length: $\|\xi\| = 1$ (i.e. on the surface of the unit hypersphere S^3). This may prove daunting for designing a quaternion integrator (see Section 5.1) and so in the future working directly with infinitesimal rotations may prove a better alternative. However, we will be dealing with this quaternion kinematic equation mostly:

$$\dot{\xi} = \tfrac{1}{2}\vec{\omega} \circ \xi, \tag{3.6}$$

where $\circ$ denotes quaternion multiplication (for details about quaternion algebra see [Sho85]). A problem that arises is in defining the generalized positions and velocities. Clearly there are 3 degrees of freedom, so the velocity has 3 components too, i.e. the angular velocity, but we now have 4 parameters for generalized rotational position. The solution is to expand from (3.6) a linear application between generalized velocities and positions called *kinematic mapping* and use it in all calculations: $\dot{\boldsymbol{\xi}} = \frac{1}{2}\mathbf{G}\boldsymbol{\omega}$ where $\mathbf{G} \in \mathbb{R}^{3\times 4}, \mathbf{G}^T\mathbf{G} = \mathbf{1}$ [BETC14].

If a point is expressed in local coordinates $\vec{r}_0$ then its world position will

[7]To my humble understanding they can prevent singularities but not gimbal lock.

be $\vec{r} = \mathbf{R}\vec{r}_0$ or $\vec{r} = \xi \circ \vec{r}_0 \circ \xi^*$, where * denotes quaternion conjugation. The linear velocity of this point in world space is:

$$\dot{\vec{r}} = \vec{\omega} \times \vec{r}. \tag{3.7}$$

If the point belongs to a body that has both a translational and rotational movement, the world position is then $\vec{r} = \vec{x}_{CM} + \mathbf{R}\vec{r}_0$ and its world velocity is:

$$\dot{\vec{r}} = \vec{v}_{CM} + \vec{\omega} \times \vec{r}, \tag{3.8}$$

where by CM we denote the center of mass.

3.4 Newton-Euler equations for rigid bodies

In order to be able to write the rigid body equations of motion we need to introduce a few more physical quantities. The most important one is the *angular momentum*, which as the name states is the analog of linear momentum: $\vec{L} = \sum \vec{r}_i \times m_i \vec{v}_i$. Using equation (3.7) we can expand it to $\vec{L} = \sum m_i \vec{r}_i \times (\vec{\omega}_i \times \vec{r}_i)$ which in turn can be reformulated as a linear mapping applied to the angular velocity: $\vec{L} = \mathbf{I}\vec{\omega}$, with

$$\mathbf{I} = \sum m_i (r_i^2 \mathbf{1} - \vec{r}_i \vec{r}_i^T). \tag{3.9}$$

The symmetric matrix $\mathbf{I}$ is called the *inertia tensor* and it is the analog of mass for rotations. You can find more details about the inertia tensor, its properties and transformation under rotation, as well as ways to compute it for different shapes in Section 5.1 and the references [GPS02, Bar97].

The Newton-Euler equations in aggregated Cartesian coordinates are:

$$\tilde{\mathbf{M}}\dot{\mathbf{v}} = \mathbf{f}(\mathbf{x}, \mathbf{v}), \tag{3.10}$$

$$\mathbf{I}\dot{\boldsymbol{\omega}} = \boldsymbol{\tau} - \boldsymbol{\omega} \times \mathbf{I}\boldsymbol{\omega}, \tag{3.11}$$

where $\tilde{\mathbf{M}} = \mathrm{diag}(m_i)$, $\boldsymbol{\tau}$ is the external torque and the relationship between linear and angular velocities (and corresponding generalized positions) is

given by the kinematic equations. The last term in (3.11) is the Coriolis term and it is often left out in game physics engines; this is because it is a nonlinear term (quadratic in velocities) and can become stiff and cause energy gain under explicit integration [Cou14]. We have not investigated this issue in this thesis but it has to be kept in mind as it affects the conservation of angular momentum.

3.5 Continuum mechanics

Continuum mechanics is a too broad field to be summarized here, but we will do our best. The most important quantity to measure deformation is *strain* which is analogous to relative spring elongation. In order to be able to define strain we need to introduce the deformation field and its gradient. Thus for every point in the original undeformed object $\vec{r}$ there corresponds a displaced point $\vec{x} = \phi(\vec{r}) = \vec{r} + \vec{u}(\vec{r})$, where u is the displacement. The gradient $\mathbf{F} = \nabla\phi$ indicates the amount of deformation and also of accumulated elastic energy [SB12]. Given that $\mathbf{F}$ is a linear mapping that approximates the deformation locally it is natural to define strain as the deviation of the matrix from a pure rotation [MSJT08], i.e. shearing and scaling. We measure this by seeing how far off is $\mathbf{F}$ from being orthonormal and we obtain the Green-Lagrange nonlinear strain:

$$\boldsymbol{\varepsilon}_G = \tfrac{1}{2}(\mathbf{F}^T\mathbf{F} - \mathbf{1}). \tag{3.12}$$

To be more precise, strain is a rank 2 tensor (basically a 3 by 3 matrix) which is also symmetric by equation (3.12). Nonlinearity is often discarded by using the linearized Cauchy strain:

$$\boldsymbol{\varepsilon}_C = \tfrac{1}{2}(\mathbf{F} + \mathbf{F}^T) - \mathbf{1}, \tag{3.13}$$

but this only works for small deformations.

Now that we know about strain, we can move on to the *stress* tensor which is the analog of force. Together with strain they form an analog of Hooke's law:

$$\boldsymbol{\sigma} = \mathbf{C}\boldsymbol{\varepsilon}, \tag{3.14}$$

where $\mathbf{C}$ is rank 4 symmetric tensor defining material properties. Due to symmetry both the strain and the stress can be turned into column vectors using Voigt notation [MSJT08] and $\mathbf{C}$ becomes a 6 by 6 matrix: $\tilde{\boldsymbol{\sigma}} = \mathbf{C}\tilde{\boldsymbol{\varepsilon}}$. Also given material isotropy the number of parameters describing elastic properties is two: the Young's modulus E and the Poisson ratio ν (or alternatively the Lamé coefficients). The stiffness matrix is then:

$$\mathbf{C} = \frac{E}{(1+\nu)(1-2\nu)} \begin{bmatrix} 1-\nu & \nu & \nu & 0 & 0 & 0 \\ \nu & 1-\nu & \nu & 0 & 0 & 0 \\ \nu & \nu & 1-\nu & 0 & 0 & 0 \\ 0 & 0 & 0 & 1-2\nu & 0 & 0 \\ 0 & 0 & 0 & 0 & 1-2\nu & 0 \\ 0 & 0 & 0 & 0 & 0 & 1-2\nu \end{bmatrix}, \tag{3.15}$$

where ν is between 0 and 0.5 and E has the dimension of pressure.

The equation (3.14) only holds for linear elasticity using Cauchy strain. This is too limiting for computer graphics where we need to view large deformations. This is why we use the St. Vennant-Kirchoff model and the Green-Lagrange strain. In this case the stress tensor (also called the first Piola-Kirchoff tensor) becomes: $\boldsymbol{\sigma} = \mathbf{FC}\boldsymbol{\varepsilon}$. But the two formulations can be unified by using a energy density formulation of the constitutive model:

$$\Psi(\mathbf{F}(\vec{x})) = \boldsymbol{\varepsilon}^T \mathbf{C}\boldsymbol{\varepsilon}. \tag{3.16}$$

The total energy is the integral of Ψ over the whole volume domain and the stress is $\boldsymbol{\sigma} = \frac{\partial \Psi}{\partial \boldsymbol{\varepsilon}}$. There are also other models for elastic continua like the co-rotational model or the neo-Hookean one [SB12].

The equation of motion for continuous media is:

$$\rho \frac{\partial^2 \vec{x}}{\partial t^2} = \nabla \cdot \boldsymbol{\sigma} + \vec{f}_{ext}, \tag{3.17}$$

where ρ is the material density, $\nabla\cdot$ is the generalized divergence operator and $\vec{f}_{ext}$ is the external force. Note that the position term is actually a vector field, i.e. a function of both space and time $\vec{x}(\vec{r}, t)$, and this makes the

whole equation a *partial differential equation* (PDE). For more details you can consult one of the textbooks [BW97, Bat06, ZTZT77].

Chapter 4

Time discretization

Time discretization means basically numerical integrators for ODEs. They are the heart of a simulator as they are the only way of advancing trajectories in time (by a given time step). Even without constraints they form an entire field of study and lie at the foundation of many complex simulators used in physics, chemistry and engineering. Integrators for PDEs are a whole different subject and have many extra subtleties especially in the context of finite differences (e.g. fluid simulations) and finite elements. We do not treat them explicitly in this thesis as we rely on the fact that the finite element method is basically a conversion to ODE [SSB13] and so we can use familiar integrators.

In this chapter we briefly introduce some notions about numerical integration and introduce the implicit Euler integrator which is the workhorse of this thesis. We mention the basic ideas of variational and symplectic integrators and then introduce the minimization form of implicit Euler. In the end we show how this minimization can be solved in a novel way using the nonlinear conjugate gradient algorithm.

4.1 Numerical integration

There is an extensive literature on numerical integration of ODEs [HNW87, PTVF07]. They usually apply to first order equations but as we have seen in

the previous chapter we can convert the second order equations of motion to first order. Still, there are methods (like Verlet [Ver67]) that apply directly to the second order equations.

We are usually interested in two properties of the integrators: accuracy and stability. Accuracy is generally denoted by the order of the integrator, meaning the power of the time step to which the error is proportional. In this thesis we do not use order higher than two. Stability is a more delicate issue and refers to keeping the error bounded over a long integration time. There are two big classes of integrators: explicit and implicit, and only the latter can guarantee unconditional stability.

In general a first order ODE has the form:

$$\dot{x} = f(x, t), x(t_0) = x_0. \tag{4.1}$$

The numerical integration of this equation uses a time step h and has the general form:

$$x^{l+1} = \Lambda(x^l, f, h), \tag{4.2}$$

where the super-script denotes the current simulation frame, i.e. l corresponds to time $t_l = t_0 + lh$ and usually $t_0 = 0$. We only work with fixed time step in this thesis, although there are many methods that can use variable time step size and are even adaptive in order to further reduce the integration error.

The difference between one-stage and multi-stage methods as well as explicit and implicit lies in the points where the derivative f is evaluated. One stage explicit methods only need $f(x^l)$ while multi-stage methods require more points (e.g. RK4 uses 4 points). Implicit methods evaluate f in x^{l+1} or intermediary points between x^{l+1} and x^l. The challenge of implicit integration is that equation (4.2) becomes implicit (the unknown appears on both sides) and is most of the time nonlinear. Thus, implicit integration usually involves solving a nonlinear equation with numerical schemes like Newton's method.

Stability is hard to describe in brief and so we refer the reader to a couple of references [Str07, WH91]. The main idea is that the variables of the

system should remain bounded for all the period of integration; this should be also true for the total energy in dynamics. This can only be achieved in certain narrow domains of stability for explicit integrators, while implicit integrators are unconditionally stable for any time step size [Hau05]. Explicit integrators gain energy and explode numerically and implicit integrators artificially dissipate energy. Currently there is ongoing research on developing integrators that conserve as much as possible the energy and other quantities despite the integration error (see Section 4.2).

Let us illustrate the some of these concepts with the very simple and popular explicit Euler method:

$$x^{l+1} = x^l + hf(x^l). \tag{4.3}$$

The *local truncation error* is given by the residue term in the Taylor series approximation, i.e. the difference between the real solution and the computed one. It is easy to see here that it is $O(h^2)$. The *global error* is the accumulated local error over the number of steps and can be shown to be $O(h)$. This is what we usually call the integration error and it also gives the order of the integrator (the higher the better). Explicit Euler is a first order integrator so it has the worst possible error. There exist similarly complex integrators of order two (e.g. Verlet, Leapfrog), but in general higher order integrators are increasingly expensive. Among the traditional explicit integrators we mention the Runge-Kutta family, e.g. RK2 or trapezoid, RK4 and so on.

The stability bound of explicit Euler is given by $h < 1/\rho(\mathbf{G})$ where $\mathbf{G}$ is the iteration or growth matrix, i.e. $x^{l+1} = \mathbf{G}x^l$. The spectral radius $\rho(\mathbf{G})$ is usually in close connection to the highest frequency of the system. Thus, if the highest mode of oscillation exceeds the by a constant factor the "sampling" frequency of the time step, then the system will start gaining energy for that mode and amplify its oscillations. Implicit Euler uses $f(x^{l+1})$ instead and this implies that energy will always decrease down to zero no matter what time step is used. As a note, the amount of dissipation for implicit Euler is proportional to the time step size. The unconditional stability property of implicit Euler makes it a very popular integrator in computer

graphics [BW98, MSJT08]. It allows for quite large time steps compared to explicit integrators and it is efficient despite its bigger cost. Also, the artificial damping property is seen by many as a helpful feature as most phenomena in nature have some form of dissipation. This is especially convenient in computer graphics where experimental validation is not required.

As we will see, implicit Euler (or Backward Euler) is the integrator of choice in our thesis too. You may be surprised by the very low order of the integrator, but in computer graphics it is quite common to trade accuracy for speed whilst the simulation remains robust. Also, nonsmooth time stepping integrators cannot exceed first order accuracy [KP12]. Still, one can use if desired more complex implicit integrators: BDF-2 [CK02, EB08] (and other multi-step backward difference methods), Newmark [KMOW99, SSB13], Crank-Nicolson/Implicit Midpoint [Str07, Lac07], HHT (Hilber - Hughes - Taylor) or generalized α.

The equations of motion (3.2) can be discretized using the Backward Euler integrator in the following way (among others):

$$\mathbf{M}\Delta\mathbf{v} = h\mathbf{f}(\mathbf{x}_0 + \Delta\mathbf{x}), \tag{4.4}$$

$$\Delta\mathbf{x} = h\Delta\mathbf{v}, \tag{4.5}$$

where $\Delta\mathbf{v} = \mathbf{v}^{l+1} - \mathbf{v}^l$ and $\mathbf{x}_0$ is an initial point such that $\mathbf{x}^{l+1} = \mathbf{x}_0 + \Delta\mathbf{x}$.

4.2 Variational and symplectic integrators

The subject of variational or geometric or symplectic integration is currently under development by mathematicians. The three words have almost similar meaning and originate from properties of analytical mechanics: the variational principles of mechanics [Lan70], the geometric view on mechanics [HLW06] and the symplectic property of Hamiltonian phase space [Arn13, LR04]. In a nutshell, variational integration starts from the very basic principles of mechanics and develops a subset of integrators that fit with the nature of dynamical systems. The main goal is to conserve energy and momentum as much as possible irrespective of the integrator order. And

this is achieved best by so-called symplectic integrators that preserve structure over the phase space flow. Simple examples include (Størmer-)Verlet, Symplectic Euler, Leapfrog [HLW03] and others. There are also integrators that can achieve full energy conservation like Implicit Midpoint but still this is very hard to achieve in practice and may require a variable time step [Str07, Lac07].

For a detailed treatment see the subject of *discrete mechanics* developed among others by Marsden and his collaborators [MW01], the monograph of Hairer and Wanner [HLW06] or the thesis of Lacoursière [Lac07]. We are not giving too much detail here as variational integrators are not our main work horse. We use them though for two goals. The first one is to show that implicit Euler can be replaced by the similar Newmark scheme (which is variational [KMOW99]), thus reducing the amount of artificial dissipation. The second goal is to show the variational origins of the minimization structure of integrators and constraint solvers. Hamilton's principle thus gives us the base idea for extremizing a certain quantity (i.e. the Lagrangian) and discrete mechanics makes the connection to the time step slice where we need to minimize the sum of kinetic energy and potential energy.

4.3 Integration as minimization

Relatively recently the Implicit Euler integrator was recast as a minimization problem [LBOK13, BML+14, MTGG11]:

$$\text{minimize } \tfrac{1}{2}(\mathbf{v}^{l+1})^T\mathbf{M}\mathbf{v}^{l+1} + V(\mathbf{x}^{l+1}). \tag{4.6}$$

It can also be expressed in terms of positions or displacements (as it is done in the references) if making use of equation (4.5). We will do this and go in more detail about using this integrator when switching to constrained systems (Section 6.5). However, at this point we can introduce our first contribution, which is to apply a nonlinear minimization algorithm directly on (4.6) instead of solving the nonlinear optimality conditions with Newton's method [PTVF07] or in a linearly implicit fashion as it was customary

[BW98]. We used a Nonlinear Conjugate Gradient (NCG) algorithm [She94] and found that we could ignore the second derivative of the potential in a stable and accurate manner.

We now present a derivation of the algorithm that can also be found in Section 3 of [FM15a]. Not all the concepts for understanding it have been introduced but nevertheless the reader can come back later to it. For more details about modeling cloth to which this solver is applied to see Section 5.4. Also, most of the theory here is more related to the implicit integration of stiff elastic systems that is found in a big part of the simulation literature [BW98, Hau05], but not so much in constrained dynamics (the focus of this thesis). We will draw more parallels between the two approaches in the following chapters (see Section 6.8 for example).

Let us introduce first the linear elastic force $\mathbf{f} = -\kappa \mathbf{c}(\mathbf{x}) \nabla \mathbf{c}(\mathbf{x}) = -\nabla V$, where $\mathbf{c}(\mathbf{x})$ is the elongation in the springs (or constraint function) and κ is the stiffness value. If we consider the initial candidate state consisting of $\tilde{\mathbf{v}} = \mathbf{v}^{(n)} + h\mathbf{f}_{ext}$ and $\tilde{\mathbf{x}} = \mathbf{x}^{(n)} + h\tilde{\mathbf{v}}$ we can rewrite (4.4) using a first order Taylor expansion around $\tilde{\mathbf{x}}$:

$$\mathbf{M}\delta\mathbf{v} = h\left(\mathbf{f}(\tilde{\mathbf{x}}) + \mathbf{K}\delta\mathbf{x}\right), \tag{4.7}$$

where $\mathbf{K} = \nabla_{\mathbf{x}}\mathbf{f} = -\frac{\partial^2 V}{\partial \mathbf{x}^2}$ is the *tangential stiffness matrix* and $\delta\mathbf{x} = h\delta\mathbf{v}$. Most authors choose to solve the implicit integration problem using only one Newton step, meaning we only need to solve one single linear system: $\mathbf{S}\delta\mathbf{v} = \mathbf{t}$. This works well in practice, but only if $\mathbf{K}$ contains second derivatives of the constraint function. This is because these terms contain information about the change of the constraint direction, so without them we need an iterative algorithm that keeps updating the constraint gradient. By dropping the second derivative term from $\mathbf{K}$ (see [BW98]) we get:

$$\mathbf{K} = -\kappa \mathbf{J}^T \mathbf{J}, \tag{4.8}$$

where $\mathbf{J}$ is the Jacobian of the constraint function. This is equivalent to linearizing the elastic force as in [EB08]. Using this formula at every Newton

iteration we get the series of linear systems we need to solve:

$$(\mathbf{M} + h^2\kappa\mathbf{J}_k^T\mathbf{J}_k)\delta\mathbf{v}_{k+1} = h\mathbf{f}(\mathbf{x}_k), \tag{4.9}$$

where $\mathbf{J}_k^T = \nabla\mathbf{c}(\mathbf{x}_k)$ and $\mathbf{x}_{k+1} = \mathbf{x}_k + h\delta\mathbf{v}_{k+1}$.

NCG is a natural solution for solving the above problem, given its linear version is very popular for solving the one Newton step approach. The only change we need to make to linear CG is to replace the system matrix at every step with $\mathbf{S}_i = \mathbf{M} + h^2\kappa\mathbf{J}_i^T\mathbf{J}_i$ (the Hessian of the objective function) and the residual with $\mathbf{r}_i = h\mathbf{f}(\mathbf{x}_i)$. We use a Fletcher-Reeves formula and perform the inner line search in only one iteration - see Algorithm 1.

Algorithm 1 NCG implicit solver

Unconstrained step to $\tilde{\mathbf{x}}, \tilde{\mathbf{v}}$
Compute Jacobian $\mathbf{J}$ and forces $\mathbf{f}$
Compute residual $\mathbf{r} = \mathbf{d} = h\mathbf{f}$ and its square $\delta = \mathbf{r}^2$
for iter = 1:maxIter **do**
  Compute $\mathbf{q} = \mathbf{S}\mathbf{d} = (\mathbf{M} + h^2k\mathbf{J}^T\mathbf{J})\mathbf{d}$
  Compute impulse $\mathbf{p} = \alpha\mathbf{d}$, where $\alpha = \delta/\mathbf{q}^T\mathbf{d}$
  Integrate: $\mathbf{v} \leftarrow \mathbf{v} + \mathbf{p}$, $\mathbf{x} \leftarrow \mathbf{x} + h\mathbf{p}$
  Recompute Jacobian $\mathbf{J}$ and forces $\mathbf{f}$
  Compute residual $\mathbf{r} = h\mathbf{f}$ and its square $\delta' = \mathbf{r}^2$
  Compute $\beta = \delta'/\delta$ and then $\delta' \leftarrow \delta$
  Compute new search direction $\mathbf{d} = \mathbf{r} + \beta\mathbf{d}$
end for

Note that the NCG method is not necessarily faster than traditional CG linear implicit solvers (we found that it takes roughly 40% more time without optimizations). We can also add back the second derivative term if we want. Also, visually there is no big difference between the two methods. Even in terms of energy behavior NCG is very similar to both PBD and implicit methods (Figure 4.1). The only advantages you would get with the NCG method are smaller spring elongations and more stability for large time steps. But the main reason for devising the scheme is the similarity to PBD which we exploit later in the thesis.

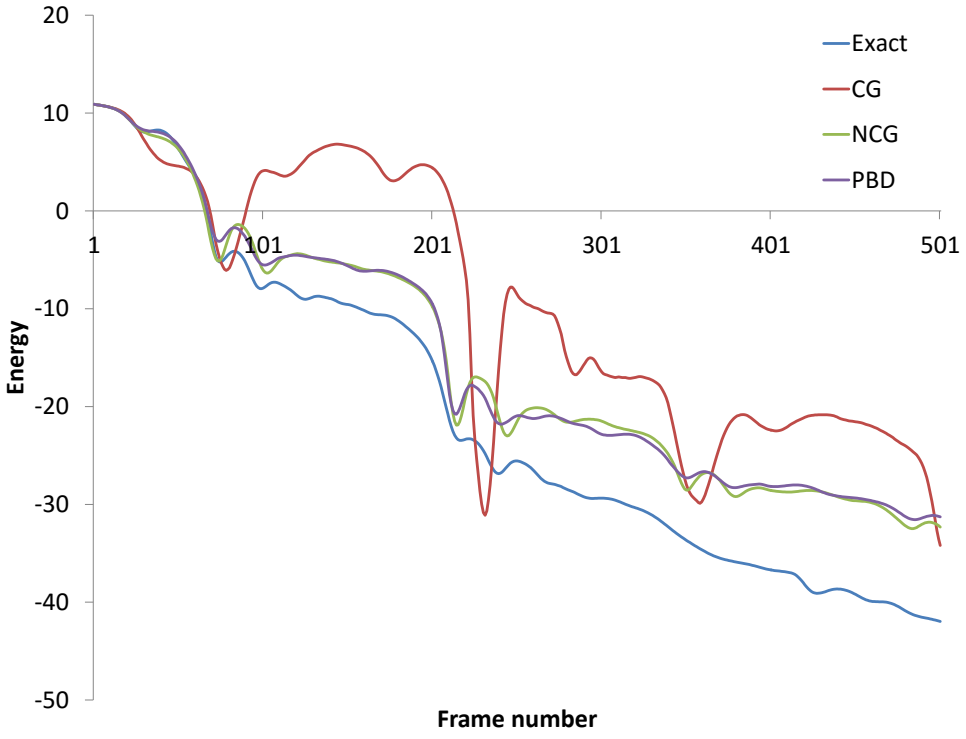


Figure 4.1: The energy evolution over 500 frames of a 15×15 piece of cloth using NCG (green), PBD (purple) and CG (red) and exact (blue) linearly implicit solvers.

Chapter 5

Material models

This chapter is in fact about space discretization. If this concept does not apply so much to particles and rigid bodies, this is not the case for deformable bodies where the choice of discretization makes a world of difference between different methods and simulation results. Even for rigid bodies we need to represent the surface geometry for collision detection, and this is often done with meshes, even if many times they can be approximated by analytical surfaces. The chapter presents itself more as a survey of previous work and no essentially new things are introduced, but it is needed as a prerequisite for understanding the next chapters and the overall results of this thesis. We also insert where relevant snapshots of some of our own simulations.

5.1 Rigid bodies

Rigid bodies are large solid objects (much larger than a particle) that have shape and a distribution of mass. As already described in Chapter 3 a rigid body has 6 degrees of freedom: 3 for translation and 3 for rotation. In practice we can use somewhere between 6 and 12 parameters to describe the generalized position of a rigid body. This is still a small footprint compared to a deformable body where many points can move relatively (and not in a rigid transformation) and this is why rigid bodies are so attractive in interactive or large-scale simulations.

The Newton-Euler equations of motion for rigid bodies were presented in Section 3.4 and we are using a quaternion representation of rotations. We now have to provide an integrator for the quaternions and we present two options (both implicit in the velocities):

$$\xi^{l+1} = \xi^l + \tfrac{h}{2}(\xi^l \circ \vec{\omega}^{l+1}), \tag{5.1}$$

followed by a renormalization of the quaternion, or

$$\xi^{l+1} = \xi^l \circ e^{(0, \frac{h}{2}\vec{\omega}^{l+1})}, \tag{5.2}$$

as described in Section 2.1 of [TA11].

We have introduced the inertia tensor in Section 3.4 without much detail. In the case of continuous materials (e.g. rigid bodies) the sum in the definition of the inertia tensor (3.9) becomes an integral [Bar97]. Usually the inertia matrix of rigid bodies used in numerical implementations are computed from analytical formulas of geometric primitives and compositions of several such matrices (through the relation between moments of inertia about parallel axes) [GPS02]. The resulting matrix is stored in the local body frame as $\mathbf{I}_b$ and then transformed to world space using the formula $\mathbf{I} = \mathbf{R}\mathbf{I}_b\mathbf{R}^T$ (the same is true for the inverse matrix). Another option is to diagonalize the matrix and obtain the principal inertia directions from the eigenvectors and use these as the initial local frame.

The resulting mass matrix for a rigid body has the form:

$$\mathbf{M}_j = \begin{bmatrix} m_j\mathbf{1} & 0 \\ 0 & \mathbf{I} \end{bmatrix}. \tag{5.3}$$

The total mass matrix of a rigid body system is a block diagonal matrix $\mathbf{M} = \mathrm{diag}(\mathbf{M}_j)$. But it can also be made out of mass matrices of both rigid bodies and particles (or FEM nodes). In practice we use the inverse of the mass and of the inertia tensor instead. This is because an immovable object has infinite mass and inertia moments; thus it easier to store zero instead and makes the calculations simpler. Also, storing the inertia tensor as a diagonal

matrix makes computing its inverse easier.

Even if we do not introduce yet formally the concept of constraint, it is natural to understand what it means and how it describes interactions between rigid bodies. Most often bodies are articulated using joints: translational or rotational joints, which remove a certain number of degrees of freedom. Most popular joints are [Smi06, Mir96]:

- *prismatic* or slider - allows only one dimensional translation;
- *spherical* or ball-and-socket - allows rotation, but not translation;
- *revolute* or hinge - has only one rotational degree a freedom (around a given axis);
- *universal* or Cardan - two rotational degrees of freedom left.

The rotational joints can also have angle limits. For an example of simulation of bodies articulated using ball-and-socket joints see Figure 5.1.

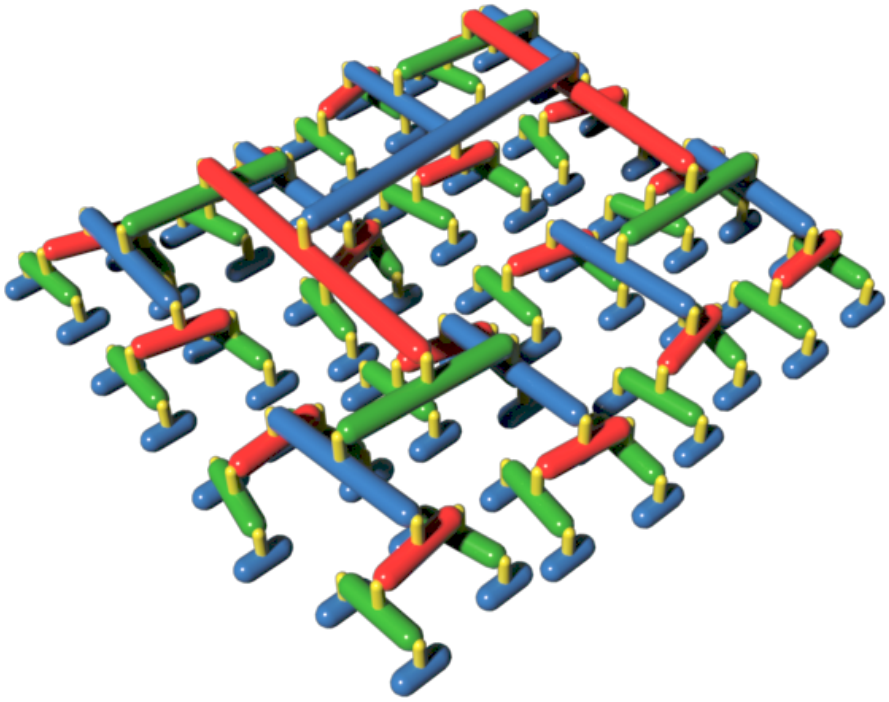


Figure 5.1: Hierarchy of articulated rigid bodies with spherical joints.

Another type of interaction between rigid bodies is contact, also known as collision or impact. We will give a different meaning to each of these words. By contact we will mean a type of constraint that prevents bodies from interpenetrating (especially when they are resting on top of each other). Contact is mostly persistent, manifesting friction in the tangential plane (stick), but it can also break (slip). Collision is a word that we will reserve mostly for the phase of collision detection, as it has caught on in time and it refers to testing for intersection between objects and obtaining contact information. Impact refers almost exclusively to high speed collisions that usually result in the objects moving apart immediately after (i.e. elastic or partly elastic impacts). Fully inelastic impacts are better handled by contact constraint approaches as penetration causes unforeseen problems. Impacts are generally governed by *impact laws* which are expressed in terms of velocities. There are two famous impact laws: Newton - where the outbound normal velocity is a fraction of the inbound one and Poisson - which splits the impact in two phases (compression and expansion) and there is a ratio between the two corresponding impulses. Simultaneous impacts of more than two bodies are more difficult to analyze and the aforementioned impact laws can lead to energy increase. This is why new impact laws have been derived and also there is ongoing work on better accommodating reflections into the constraint framework [Ste00, SKV$^+$12].

The problem of contact is more general than the context of rigid bodies as it also encompasses the rest of bodies presented in this chapter. This is why we dedicate the entire Chapter 7 to it. We just want to briefly mention now that there are two main approaches when dealing with contact in computational mechanics: the penalty approach and the Lagrange multipliers approach. Both approaches can be found in relatively old works, although we are not sure when they first appeared. We can say at least that the penalty approach started with the work of Hertz (1882). It is unclear though when Lagrange multipliers were first used for unilateral constraints. Even the names suggest that from the start professionals realized that these were just two ways of solving a constrained optimization or variational problem [WL06]. We are going to use this argument to show that the methods are not

so different from each other, even if they have grown quite separate over the decades. Another tricky issue is that of Coulomb friction which exhibits a lot of problems: e.g. stick-slip transitions, normal-tangential force coupling. This becomes especially apparent in constraint formulations is spite of the simple friction laws. Again, all of these will be handled in a nonsmooth dynamics setting in Chapter 7. As a preview, see in Figure 5.2 a simulation of rigid boxes piling up using our new position based solver.

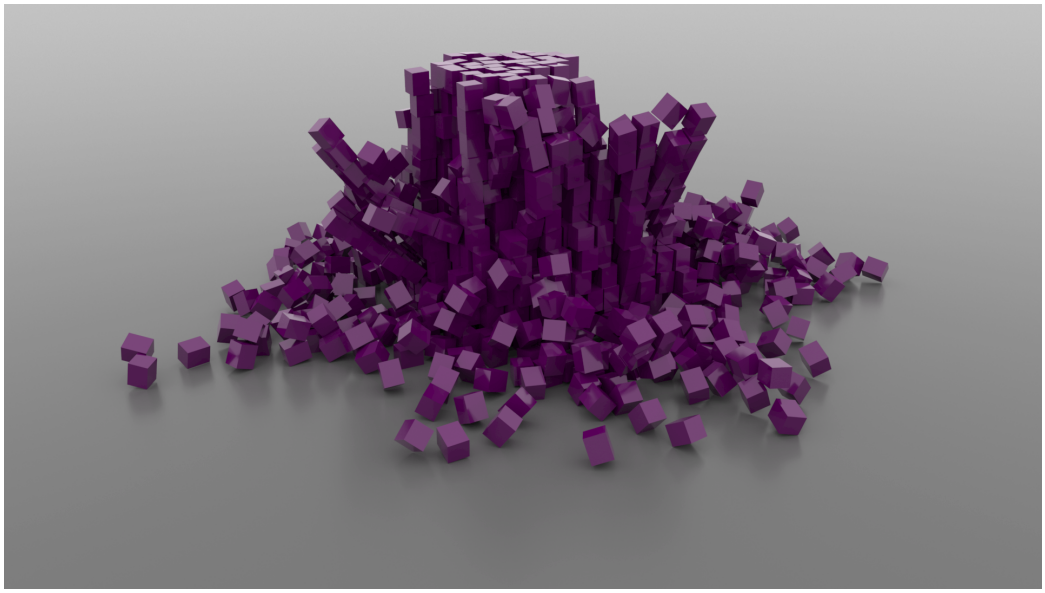


Figure 5.2: Boxes falling on ground solved with position projection.

5.2 Elasticity

In this section we address elastic materials. Bodies made of such materials are called (at least in our context) deformable, flexible or soft bodies. There are many ways to model deformable bodies and they all involve some kind of approximate spatial discretization. Just like in the case of integration in time, the accuracy depends on the size of the discretization resolution.

One of the most popular methods of discretization used in computer graphics consists of particle systems interconnected by linear springs or other forces. This approach is at the heart of our cloth simulations as they have

been widely used in the past and have proven very effective. Unfortunately they are not the most accurate or physically correct. But they are quite close and look very plausible. In the next three sections we will present mass-spring approaches for threads, cloth and volumetric soft bodies.

The other two popular approaches to modeling elastic bodies are finite differences and finite element. There exist others too, like finite volume, boundary element, mesh-free methods but they were not studied here.

5.3 Threads

Threads are most often modeled as a chain of particles connected through springs. These springs can be either integrated implicitly or treated as hard links in a constraint approach. More complex models involve articulated bodies [Had06], Cosserat theory [Pai02], supper-helices [BAC+06] and nons-mooth dynamics [BDCDA11]. In addition, threads need modeling of torsion [KPGF07, Had06, CC13].

The mass-springs model derivation for threads is actually the same road that was taken to derive the string vibration equation but in reverse. We show now how starting from the wave equation can explain the choice of the particle mass and of the spring stiffness:

$$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial u}{\partial x}, \tag{5.4}$$

where u is the transversal displacement (longitudinal waves are similar) and c is the wave propagation speed. One can inspect any derivation of this equation (for example the Wikipedia page) and notice that it starts from a chain of N particles of mass m and springs of constant k (see Figure 5.3). The total mass then corresponds to $M = Nm$ and the total stiffness to $K = k/N$ so that in the end we get $c^2 = KL^2/M = E/\rho$, where $L = N\Delta x$ is the string length, E is Young's modulus and ρ is density. It follows that the particle mass and the spring constant need to be tuned according to the number of particles in order to get the same behavior of the discretization. This mass-spring view can also be seen as a finite difference approach to solving the PDE

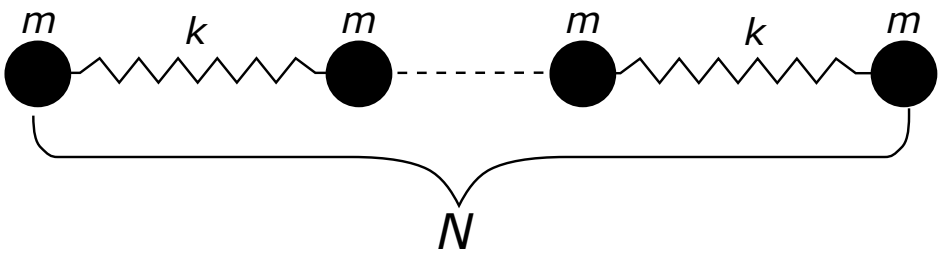


Figure 5.3: Mass-spring approximation of a vibrating string.

in (5.4) [Str07]. Although in two or three dimensions springs no longer suffice and need to be replaced by stress-strain relations, this is a good analogy to keep in mind for the remaining of the thesis, especially when working with deformable bodies modeled as mass-spring systems.

In computer graphics applications of threads involve mainly simulation of hair [HH12, KTS+14]. Other applications include surgical threads (involving suturing and knots) or even plain ropes in VFX or games [Kny10].

5.4 Cloth

Cloth is a generic name we give in computer graphics to thin surfaces or shells or membranes with applications mainly in garment simulation. There is a whole branch of study in mechanical engineering about this type of objects [Bat06], but we are not entering this kind of detail. We focus mainly on the mass-spring approximation and at the end offer an accurate alternative using FEM.

As you can see in Figure 5.4 there are mainly 3 types of springs or links inside cloth. The stretch links are structural as they hold the fabric together and also determine its extension and compression properties, i.e. its tensile strength. Many authors prefer to model these springs as bi-phasic (different stiffness over a certain deformation), make them behave differently at compression (e.g. very low or no compression resistance) or use nonlinear forces [BHW94]. Shear links have the role of preventing skewing in the plane of the cloth; they are only an approximation of shear stresses that arise in continuous materials. Bending links join more distant neighbors and, as the name

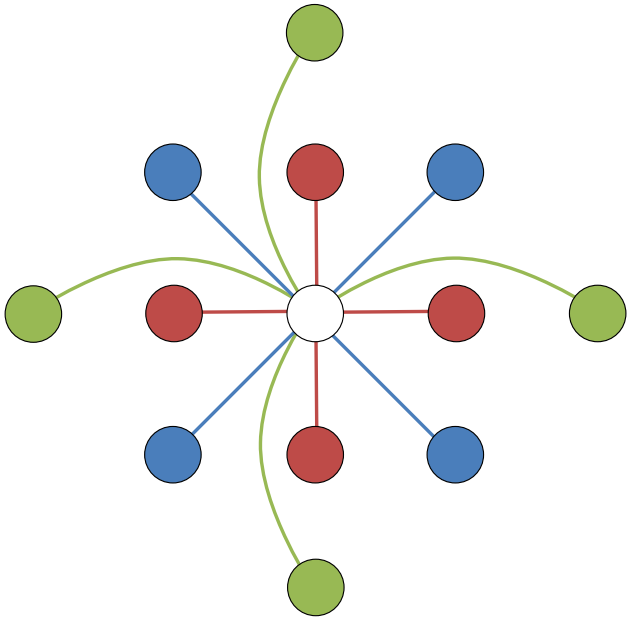


Figure 5.4: Links structure of cloth relative to one particle (white one in the center): stretch links (red), shear links (blue), bend links (green).

says, they prevent bending of the cloth (out of plane deformation). Bending links can be replaced by other type of deformation functions (e.g. dihedral angle [BW98, MHHR07], curvature [KNE10]) or potentials [BWH+06].

Cloth materials are usually very resistant to stretching and slightly less to shearing. The most freedom a fabric has is for out-of-plane movement, i.e. bending. Thus an accurate bending model is important for replicating real cloth. Other phenomena like anisotropy, hysteresis, plasticity, damping and viscosity are also important for cloth modeling [VMT00], but each introduces further complexity and performance issues. Also, depending on the model chosen, it is not trivial to set the parameters of cloth so that they match a real material (e.g. silk, denim, linen, cotton, polyester): some have to be chosen from measured parameters, some by hand and some from pre-stored presets that can be modified.

There are two ways of building a cloth simulation mesh. The first one is to start from a rectangular grid of points and set the existing edges as structural springs. Next the shear links are chosen as the diagonals of the quads and the bending links connect particles over two edges (skipping the particle

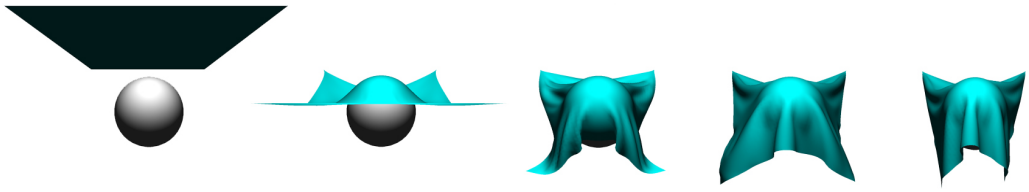


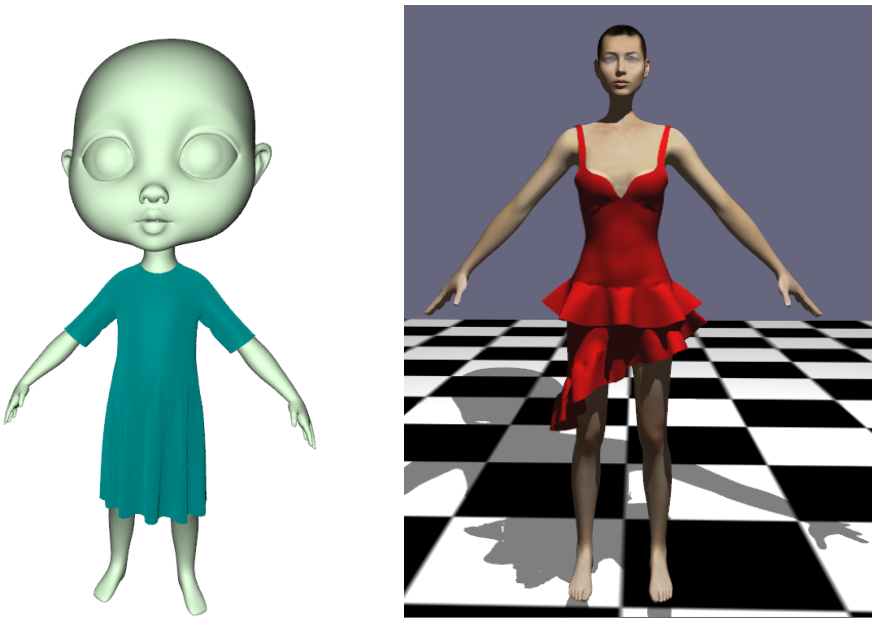
Figure 5.5: 100×100 piece of cloth hanging from corners and falling over a sphere; simulated in real time at 60 Hz using the conjugate residuals solver.

in the middle). The result is similar to Figure 5.4 for each particle; see our simulation of such a piece of cloth in action in Figure 5.5. The second method is to start from a given triangle mesh. However, this mesh should be manifold, non-degenerate and somewhat homogeneous (i.e. equally sized triangles). This time there is no difference between stretching and shearing springs, only bending ones can be constructed by connecting second order neighbors (over two edges). Using rigid links with triangle meshes usually leads to locking artifacts (the removal of too many degrees of freedom) which can be alleviated in two ways: by softening the constraints [MHHR07, FM15a], by using a continuous approach [BW98, MCKM14] or through conforming meshes [EB08]. You can see a couple of our simulations using triangular meshes and soft constraints in Figure 5.6.

Even though cloth was simulated in the past using finite differences, the most popular way of modeling cloth accurately nowadays is the finite element method [VMTF09, TWS07]. The continuum formulation in [BW98] is actually a particular case of the finite element method. The three constraints correspond to the strain components $\boldsymbol{\varepsilon}_{uu}$, $\boldsymbol{\varepsilon}_{vv}$ and $\boldsymbol{\varepsilon}_{uv}$ that make up the planar symmetric Green-Lagrange strain tensor:

$$\boldsymbol{\varepsilon}(\mathbf{x}) = \tfrac{1}{2}(\nabla\mathbf{w}\nabla\mathbf{w}^T - \mathbf{1}), \tag{5.5}$$

where $\mathbf{w} : \mathbb{R}^2 \to \mathbb{R}^3$ is a mapping from an undeformed mesh parametrization (u, v coordinates) to deformed vertex positions (corresponding in 3D to the deformation function ϕ in Section 3.5). Then the actual components of the



(a) (b)

Figure 5.6: Simulation of (a) a cloth model consisting of 6910 vertices and 13674 triangles using soft constraints and (b) a garment exported from Marvelous Designer and rendered in OGRE.

2×2 strain matrix are:

$$\varepsilon_{uu} = \tfrac{1}{2}(\mathbf{w}_u^T \mathbf{w}_u - 1), \tag{5.6}$$

$$\varepsilon_{vv} = \tfrac{1}{2}(\mathbf{w}_v^T \mathbf{w}_v - 1), \tag{5.7}$$

$$\varepsilon_{uv} = \boldsymbol{\varepsilon}_{vu} = \mathbf{w}_u^T \mathbf{w}_v, \tag{5.8}$$

where by the subscript of $\mathbf{w}$ we signify partial derivation with respect to to u and v. By considering strain constant over a triangle (linear FEM) we can derive simple formulas for $\mathbf{w}_u$ and $\mathbf{w}_v$ like in [BW98] or [VMTF09].

The integral of the strain energy over a triangle is (see Section 3.5):

$$V_{\text{fem}} = \tfrac{a}{2}\hat{\boldsymbol{\varepsilon}}(\mathbf{x})^T \mathbf{C}_{2D} \hat{\boldsymbol{\varepsilon}}(\mathbf{x}), \tag{5.9}$$

where a is the area of the triangle, $\hat{\boldsymbol{\varepsilon}} = (\varepsilon_{uu}, \varepsilon_{vv}, \varepsilon_{uv})$ and $\mathbf{C}_{2D}$ is a matrix that depends on the Young modulus E and the Poisson ratio ν (or equivalently on

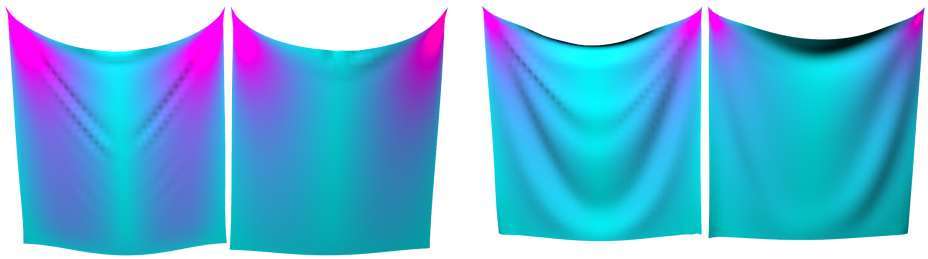


Figure 5.7: Two snapshots of a side by side real-time simulation of two 40×40 cloth pieces with the same Young's modulus E: regularized FEM constraints (left) and soft links (right); superimposed in purple is the strain map. FEM offers more realistic folds and the strain is better distributed throughout the cloth.

the Lamé coefficients) like the one given in [VMTF09] or [TWS07]. Note that the former reference expresses isotropic elasticity while the latter expresses orthotropic elasticity, i.e. different stiffness along warp and weft directions.

We used the above formulation to implement a constraint based FEM solver for cloth using the regularization framework that we will introduce in Section 6.11. You can find more details in [FM15a] and an illustration of our method in Figure 5.7.

In practice one has to make a trade-off between accuracy and speed, depending on the computing power available. Another factor that impacts both performance and realism is tessellation: coarser simulation meshes run faster but produce less folding and wrinkles. Such high frequency details can be added after the simulation in a plausible fashion [KGBS11]. There also exist adaptive tessellation techniques [NSO12] that refine the simulation mesh depending on the current level of detail, visible parts etc. Other methods involve "cheating": the simulation is pre-computed in all sorts of possible ways that the avatar could move and the new poses and animations are interpolated [KKN+13].

5.5 Virtual try on for clothes

In [FM15b] we also studied the various aspects of creating a virtual fitting room for clothes. The technologies needed for a virtual try on (VTO) system overlap a lot those required by CAD software for designing apparel [LZY10]. Some of the companies that have developed such CAD systems include Optitex, Lectra (Modaris), Toray-Acs, Gerber (AccuMark), Investronics, Assyst-Bullmer, DressingSim. Another example is the software Marvelous Designer from CLO Virtual Fashion, which is used both in the textile industry and in entertainment.

The most well-known and documented experiments with VTO are those of MiraLab in Switzerland [MTKV+11, MT10]. They identify three most important modules of such a system: a body sizing module, a motion retargeting module and a cloth simulation module. The garments are always considered available in their studies from sewing together 2D patterns around a body, just like in CAD systems.

At the moment there are no working virtual try on systems available on the market but there has been ongoing work for more than a decade by companies and academia. For example, HumanSolutions participated in such a project with several German universities [DTE+04]. Other startups in the field include PhiSix, TriMirror, Fitiquette, 3D-a-porter and possibly others.

From our standpoint there are a few key challenges that are very similar but not always to those outlined by the references from MiraLab:

- *avatar creation* - can be done through parametric interpolation via a UI, but scanning and photos could still be alternative sources;
- *avatar animation* - can be created by hand, from motion capture and there is also the option of real-time skeletal data from a Kinect like device ("virtual mirror");
- *garment acquisition* - we cannot always rely on patterns so we need to be able to reconstruct the model from existing physical clothes;
- *cloth simulation* - this is the main performance bottleneck (especially

on the collision detection side); ideally it has to be both realistic and run in real time (a middle ground has to be reached);

- *rendering* - this is preferably done in web browsers (WebGL) and on mobile (OpenGL ES); using an existing engine is recommended;
- *cloud processing* - this is a possible solution for offloading some of the simulation computation; challenges include geometry or video streaming and interaction lag.

There exist four major ways of 3D reconstructing the body: from laser scans, from structured light, from depth cameras and from photos. The most accurate body reconstruction involves laser scanning and many times it is not fully automated, requiring some human input [LZY10]. Another alternative is the TC2 body scanner using structured light (i.e. bands or other patterns of light that get deformed when projected on surfaces). Depth cameras have been used in the recent years for scanning and reconstructing objects. Commercial software for Kinect based reconstruction also exists, e.g. Skanect (Occipital) or MyBodee (Styku). Bodymetrics offers a reconstruction booth for in-shop use. Reconstruction from images is also possible, e.g. from 4 orthogonal directions photos. This is similar to the general purpose method provided by Autodesk 123D Catch. Other similar stereo computer vision based reconstruction software packages include Agisoft PhotoScan, Pix4D or VisualSFM.

For reconstructing clothes again laser scanning is the best choice, but depth cameras can be used too. Recently methods using photogrammetry and light fields have given good results; structure from motion (videogrammetry) is also possible. One preferred method for reconstructing deformable surfaces from multi-view photography is the one in [BBH08]. Capturing moving cloth techniques are described in [BPS+08] and [WCF07]. The former needs no markers, uses 16 viewpoints, but needs to fill holes. The latter needs a color pattern on the cloth and uses multiple synchronized video cameras too.

The type of the material is also important for the look of the simulation. One needs to make the difference between linen, fleece, satin, knit, silk, denim

etc. The material parameters can be obtained through direct mechanical testing of the fabric, e.g. the Kawabata Evaluation System (KES) [MT10]. A more advanced method that uses both mechanical testing and 3D cloth surface reconstruction was developed recently [MBT+12]. Others have tried to deduce these parameters from video sequences (see [BTH+03] or more recently [KNM10]).

5.6 Deformable bodies

In computer graphics there are many ways in which deformable bodies can be "tricked". We leave aside animation techniques and focus only on dynamics simulation, but even then one can use mass-springs lattices, cloth balloons, shape matching, example based deformation and the list can continue. In engineering on the other hand, where accuracy is important, the state of the art method is FEM. This is why we are also focusing in this thesis on this method only. Our goal is to show that we can use it inside our constraint framework and we can make it as accurate as needed. FEM is actually already used quite a lot in VFX in products like the DMM plugin (especially for fracture [PO09]) or Houdini.

We are not giving the details of the method in this chapter but rather focus on the geometric side of the modeling. For FEM we need volumetric meshes (that unlike surface meshes have point inside too) and for this thesis we used tetrahedral meshes (also called tet-meshes). One could also use other types of elements like boxes (hexahedral meshes) [SSB13] or more exotic shapes. In order to obtain such a mesh one usually generates it from a closed surface mesh [BP97]. For this you need to use an utility like TetGen or NetGen and make sure first that your mesh is manifold, non-degenerate, uniform, has no holes. Also you have to pay attention to the tessellation level, i.e. the number of resulting tetrahedra, as they are computationally costly. In our experiments we have only used on the order of hundreds of elements (see Figure 5.8).

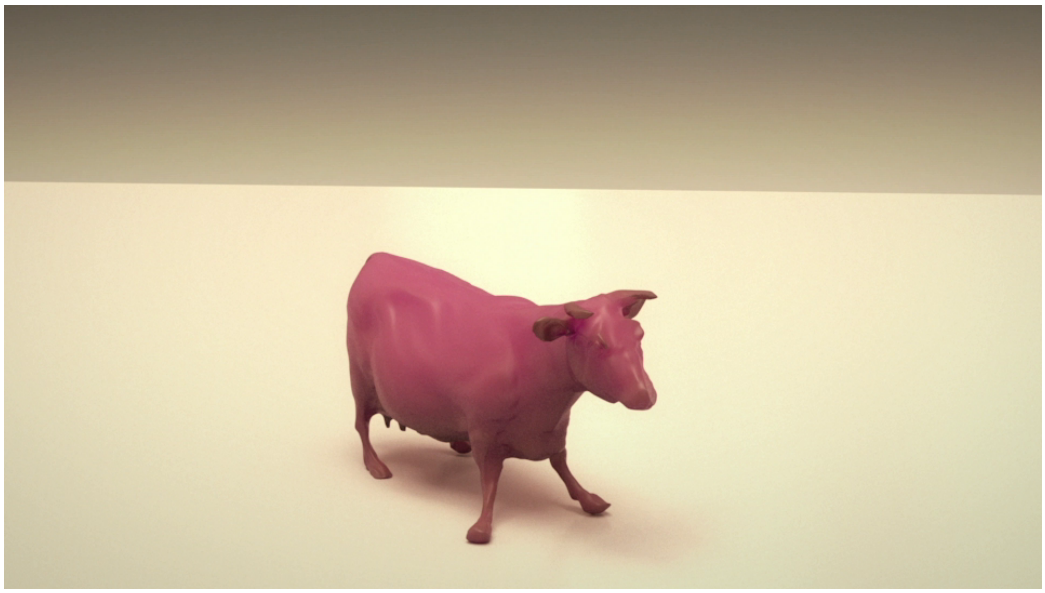


Figure 5.8: Flexible cow falling on ground.

5.7 Fluids

Fluids make up a special category of study in simulation. In engineering the field is called *computational fluid dynamics* (CFD) and it usually entails very complex and accurate methods. In computer graphics fluid simulation is also very popular and is used for a lot of special effects (VFX) and computer generated imagery (CGI). We are not dealing with fluid simulation in this thesis but we foresee ways of incorporating it in the future. This is why we give a short overview of the existing methods and point to those that are most suitable for integration with a constraint based system.

All fluid dynamics is governed by the Navier-Stokes equations:

$$\frac{D\vec{v}}{Dt} = \frac{\partial \vec{v}}{\partial t} + \nabla \cdot \vec{v} = \frac{\nabla p}{\rho} + \eta \nabla^2 \vec{v}. \tag{5.10}$$

Together with the incompressibility condition they form the Euler equations:

$$\nabla \cdot \vec{v} = 0. \tag{5.11}$$

We will not get into the details of this PDE as we are not using it in the rest

of the thesis. Still, we added it to show how similar it is to the continuous media equation (3.17). This is also the reason why many flexible materials are simulated using a very viscous fluid approach.

There are two main views in fluid simulation: Eulerian and Lagrangian. The Eulerian or grid-based view is the oldest and most widespread; it is based on the notion of field, where every quantity (scalar or vector) is a function of both space and time. Many now famous methods of fluid simulation are based on grid quantization of this fields and finite difference approximations The Lagrangian view on the other side tracks the particles of matter as they move and thus velocity is no longer a field and mass is automatically conserved. The most popular Lagrangian method is *smoothed particle hydrodynamics* (SPH). There exist also hybrid methods like semi-Lagrangian advection or the PIC/FLIP method [Bri15].

The standard SPH method has been extended in the recent years to an implicit version with better incompressibility. Two of these methods, constraint fluids [BLS12] and position based fluids are strongly connected to the formalism of constrained dynamics. This is because the incompressibility equation can be treated as a constraint for every cell with pressure being the associated Lagrange multiplier. We think that the PBF simulations in [MMCK14] are the best examples of how fluids can be coupled in the same solver with other position-based methods including the ones presented in this thesis. More recently grid based simulations were also formulated as an optimization problem (with pressure as the Lagrange multiplier) [Bri14].

5.8 Granular matter

Granular matter is a material with very curious properties as it sometimes behaves like a solid and other times like a fluid [JNB96]. For example, due to static friction piles of grains maintain their shape, but if the angle of repose is reached avalanches can be triggered. For similar reasons sand in an hourglass flows at a steady rate as the pressure does not increase indefinitely with height as in the case of hidrostatic pressure.

Granular flows are particularly suited for simulation with small rigid bod-

ies or particles in contact with friction. The size of particle granules can reach millimeters which is much larger than the atomic scales, so molecular dynamics simulations are infeasible here. Contact can be handled in two ways: the *discrete element method* (DEM) [BYM05], which is basically Hertzian contact theory, and complementarity approaches (including nonsmooth dynamics). Fluid dynamics approaches to sand simulation include grid based methods and SPH [ATO09]. The former can be extended to handle even more complex materials like snow [SSC+13].

You can see some simulations we did of granular matters using particles and position based dynamics in Figure 5.9. Also, you can find some of our first results in coupling rigid and deformable bodies (i.e. particles vs cloth) in Figure 5.10.

5.9 Collision detection

Collision detection is the process that identifies if geometrical objects overlap. It is a whole topic of research on its own. We have not focused on bringing contributions to it but did find ourselves choosing between different available methods as they affect the resulting simulation greatly. In what follows we give a short overview of collision detection techniques in order to introduce the reader to the terminology that will be used throughout the thesis. A good reference on collision detection is the book by Ericson [Eri04].

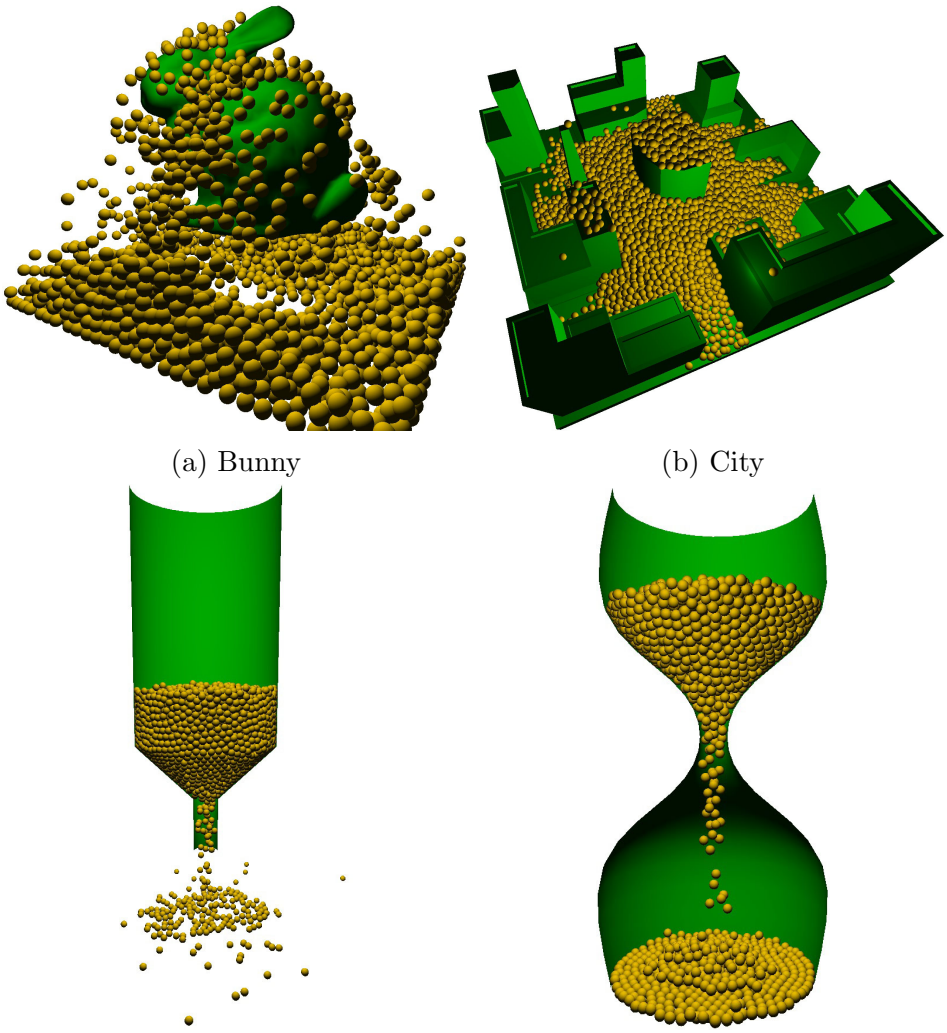
There are two important types of collision detection: discrete and continuous (CCD). The former is usually implied and it is most used as it is easier to implement. Even without the added complexity of CCD, the discrete case has many problems of its own, especially in terms of accelerating it. Discrete means that the penetration tests are done at instants of time and continuous implies that the time of impact is identified precisely in between two moments in time. CCD was developed mainly to prevent bullet through paper (or tunneling) artifacts when collisions are missed due to high speeds. Event based simulation systems may bear similarity with CCD, but in this case the simulation is not necessarily rolled back and we are only interested to catch the intersection before or when it happens. This is why CCD is mainly used

for self-intersection tests for cloth where it is very hard to tell locally which side is the correct one and there is no notion of body interior. Usually CCD techniques rely on linear sweeping of particles and shapes [Sta09, BFA02] which involve high degree polynomial equations and a lot more computational complexity. Rigid body CCD is even more complex because of the screw motion. We will talk again about CCD in Chapter 8.

Collision detection has three stages: broadphase, midphase and narrowphase. Broadphase is the stage when only potential collision candidates are identified based on simple bounding volume tests. These volumes are usually *axis aligned bounding boxes* (AABB) and one of the most popular methods used for broadphase is *sweep and prune* (SAP) [Erl05]. Midphase is optional and it usually means testing two large composite objects against one another. They usually imply using bounding volume hierarchies (BVH) for each body and doing tree traversals in order to do the overlap tests. Finally, narrowphase means doing the exact test between two relatively simple objects. Such objects usually are primitives like boxes, spheres, cylinders, capsules, triangles or convex polyhedra.

Collision detection for deformable bodies is more delicate than for rigid bodies for many reasons [TKH+05]: we cannot approximate the body with primitives, the BVH needs to be reconstructed for midphase, most models are modeled as triangle meshes and we need to account for self collisions too (see Figure 5.11). Collisions between triangular meshes is a hard aspect for any type of simulation as this popular piecewise linear discretization of geometry implies a lot of approximations and potential erroneous results. In practice a lot of spurious contact normals appear and also the resulting contact manifolds need to be filtered and reduced. Our view is that at the moment triangle vs triangle mesh collisions are sort of a black art rather than an established technology. Also, there is the unaddressed issue of the correctness of the contact points and normals and how they affect the simulation. Another unanswered question is how to handle contact patches in contrast to isolate contact point in constraint based nonsmooth dynamics. Or what about the volume of intersection which can be handled by penalty methods? Unfortunately these topics are not addressed in this thesis but

only mentioned as practical and future challenges. We end this section by mentioning an alternative technique that we have never used: *signed distance fields* (SDF), which might solve some of the above problems but we fear it may suffer from different problems, like high memory usage.



(a) Bunny (b) City

(c) Hopper (d) Hourglass

Figure 5.9: Granular matter falling over or trough various meshes.

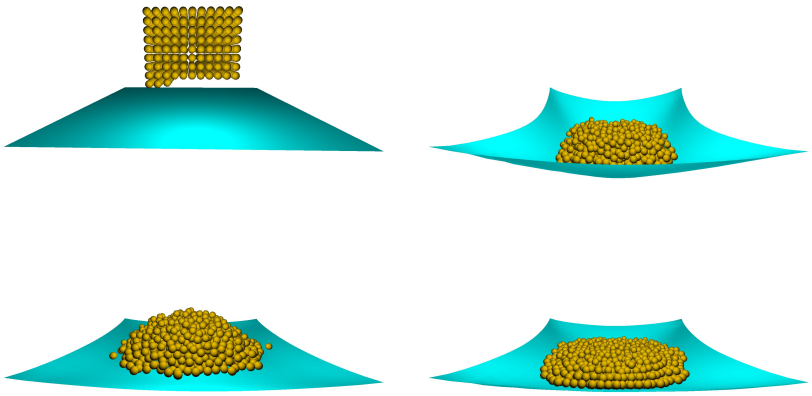


Figure 5.10: Simulation of 1000 particles falling on a 20×20 piece of cloth fixed at its corners (using the Sequential Positions method).

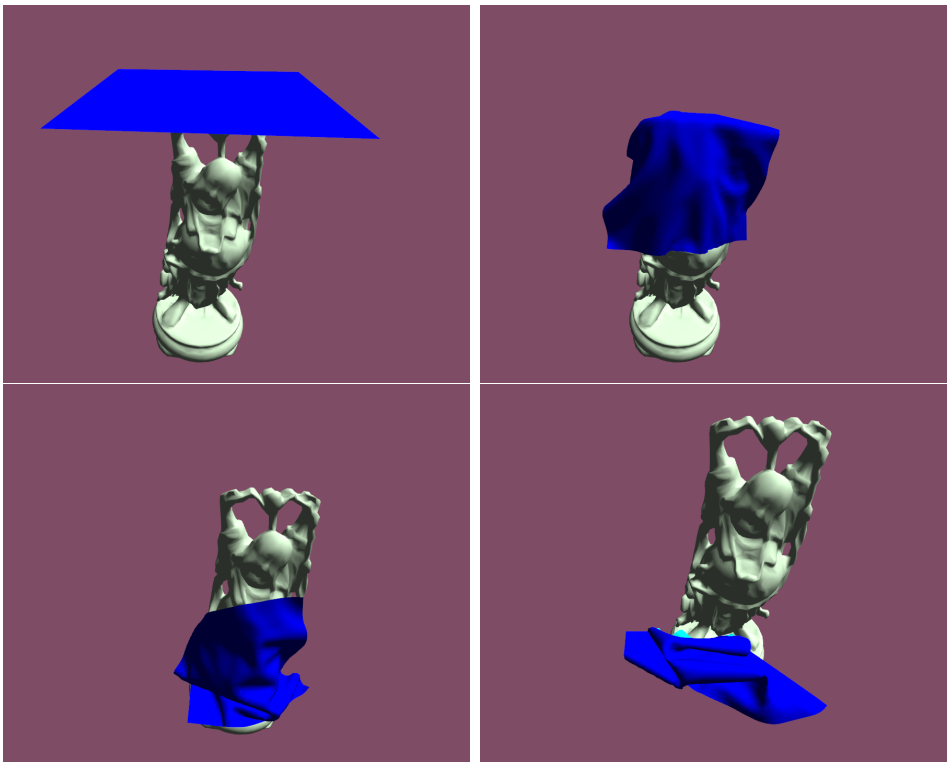


Figure 5.11: Handling of cloth-mesh collisions and self collision.

Chapter 6

Constrained dynamics

We are now starting to get at the heart of the thesis. Constraints are the stiffness limit of strong elastic forces. The method of choice for solving constraints is that of Lagrange multipliers even though it is only one side of the coin, penalty forces being the other one. We will give an extensive coverage of constraint based dynamics used in practice in order to familiarize ourselves with concepts that are often left out of textbooks (e.g. differential algebraic equations or inequality constraints). Note that our description of these mathematical tools is not thorough and many details are left out in favor of a more unifying view of current techniques.

In the end we focus mostly on solvers that are more in vein to nonsmooth or impulse based dynamics. We also go a step further by working directly at position level where we present our contributions. Note that results in this chapter are presented mainly for particle systems with bilateral constraints, although they can be extended to rigid bodies and frictionless contact too.

6.1 Constraints

We left the description of constraints for this chapter although we could have done it when describing the Lagrangian formalism (Section 3.2). Constraints usually appear in the context of analytical mechanics rather than the Newtonian one, where everything is solved by determining forces. And this is

rather surprising given that the whole point of generalized coordinates is to translate constraints into a coordinate transformation so that these constraints disappear in the end. But many times this is hard to do or even impossible[1] and constraints need to be explicitly added. This is a procedure that increases the number of variables rather than reducing it (redundant coordinates), by adding the so called *Lagrange multipliers*. It is very effective and in practice the number of unknowns can be often reduced to the number of constraints[2]. This formalism is especially helpful when working with inequality constraints. In fact our ultimate goal is to work in Cartesian coordinates and have a way of automatically computing all reaction forces in the given constraints. And this is where the Lagrangian formalism comes to help even if at first glance this was not its initial purpose.

Constraints have different classifications. Unilateral constraints involve inequality as they usually mean that a body should stay on one side of a surface at any time, i.e. contact constraint. Bilateral constraints are more popular in physics and they were given more space in the literature as they are easier to deal with than inequalities (see for example the section on Fourier's inequality in [Lan70]). Bilateral constraints have two distinctions: *scleronomic* vs. *rheonomic* and *holonomic* vs. *nonholonomic*. Scleronomic means the constraint does not depend on time explicitly while rheonomic signifies the opposite. Holonomic constraints depend only on generalized positions while nonholonomic ones involve velocities or, equivalently, position differentials. The simplest example of a nonholonomic constraint is a disk rolling on a plane [GPS02]. In this thesis however we focus mainly on holonomic and scleronomic constraints, although exceptions may appear when talking about more elaborate constraints like friction or motors. In this chapter we talk mostly about bilateral constraints and leave unilateral constraints for the end and friction for the next chapter.

[1] As is the case of nonholonomic constraints.

[2] Technically we are splitting the problem in two as the integrator still works with the unconstrained degrees of freedom. Also the number of constraints can many times exceed that of degrees of freedom if they are not independent, so this may look as disadvantageous compared to reduced coordinates formulations or penalty approaches. We thinks it is not, but this is an ongoing debate.

Generally, each constraint is a scalar function and all constraints can be aggregated into a single vector constraint function: $\mathbf{c} : \mathbb{R}^n \to \mathbb{R}^m$, where m is the total number of constraints in the system.

As hinted above, key in the understanding of constraints is the *principle of virtual work* from Lagrangian mechanics. In the case of equilibrium, the virtual displacements $\delta\mathbf{q}$ must be compatible with the constraints, i.e. $0 = \delta\mathbf{c}(\mathbf{q}) = \nabla\mathbf{c}(\mathbf{q})^T\delta\mathbf{q}$, meaning that they always lie in the tangential plane of the constraint manifold[3]. The principle of virtual work is extended from statics to dynamics by the principle of d'Alembert [Lan70]. For internal constraint forces the principle of virtual work states the following:

$$\mathbf{f}_c^T\delta\mathbf{q} = 0. \tag{6.1}$$

It follows from (6.1) that the constraint forces are perpendicular to the tangential virtual displacements and are parallel to the normal of the constraint surface, i.e. the constraint gradient $\nabla\mathbf{c}(\mathbf{q})$, and so we obtain the form of the constraint forces:

$$\mathbf{f}_c = \nabla\mathbf{c}(\mathbf{q})\boldsymbol{\lambda}, \tag{6.2}$$

where $\boldsymbol{\lambda} \in \mathbb{R}^m$ is the vector of Lagrange multipliers.

The augmented potential energy is: $\bar{V} = V - \boldsymbol{\lambda}^T\mathbf{c}(\mathbf{q})$ [Lan70]. This can be seen as coming from (6.2) or, alternatively, from a Lagrange multiplier approach to Hamilton's (or Gauss') principle with constraints added.

Another important concept is the Jacobian of the constraint function:

$$\mathbf{J} = \frac{\partial\mathbf{c}}{\partial\mathbf{q}} = \nabla\mathbf{c}^T. \tag{6.3}$$

Its physical meaning is usually the direction of the constraint force, while the corresponding Lagrange multiplier is the force magnitude. It is also the transpose of the gradient matrix which is basically a collection of column vectors representing normal vectors to each individual constraint manifold. This is why it defines the tangent space at every point and thus the whole

[3]In Cartesian coordinates this is nothing more than Newton's third law: internal forces act in pairs that sum up to zero (as the manifold is just $\mathbb{R}^n$).

tangent bundle of the constraint manifold. Therefore it comes to no surprise that we will be encountering $\mathbf{J}$ a lot in our equations.

6.2 Differential algebraic equations

The essence of this chapter is that we need to solve the equations of motion with constraints. Think of a particle falling freely in a parabola trajectory and then being constrained on an inclined plane or a bobsleigh track[4]. The trajectories are clearly different and thus the solutions to the differential equation under the same initial conditions must also be different. Such equation is some times denoted simply as a differential equation on a manifold [HLW06, LR04] and other times as a *differential algebraic equation* (DAE) [BCP96, Lac07].

The most general form of a index 3 DAE encountered in dynamics is:

$$\mathbf{M}\ddot{\mathbf{x}} = \mathbf{f}_{ext} + \mathbf{J}^T\boldsymbol{\lambda}, \tag{6.4}$$

$$\mathbf{c}(\mathbf{x}) = 0, \tag{6.5}$$

where $\mathbf{f}_{ext}$ is the external force.

An important property of the DAE is its index - a measure of singularity or perturbation: add one to the number of times it takes to derive the constraint function so that the unknown has the same derivative order as in the ODE [BCP96, HLR89]. This is in fact how DAEs of index 3 that occur in multibody dynamics are generally solved, i.e. by differentiating the constraint function twice. But other methods can use the first derivative or the original constraint function itself. These correspond (in the order they were mentioned) to working at acceleration, velocity or position level. Some methods even combine these levels, mainly due to the fact that, even if the initial conditions are compatible with the constraints, numerical methods accumulate *drift* in the constraint error at all levels. This phenomenon gave rise to the concept of constraint *stabilization*.

[4]It turns out from the solution of the brachistochron problem (that fueled the development of the variational principles of mechanics) that the fastest track is a cycloid.

In what follows we will talk about the subtleties of solving constrained systems in the context of different fields where often methods were developed in parallel giving rise to many distinct solvers.

6.2.1 Mechanical engineering

Differential algebraic equations have been developed in the context of mechanisms where many rigid parts are joined by various joints in a complex manner. Popular solvers include DASSL [BCP96], RADAU5 [HLR89] or generalized coordinate partitioning [HY90] and orthogonalization methods [MBPV11]. Index reduction is a penalty approach which converts the DAE into an ODE [Bau72]. This is basically a spring-damper regularization and it is used in modern DAE solvers as a correction technique (now called Baumgarte stabilization) [ACPR95].

Note that this is not the only possible approach to articulated systems; the others include reduced coordinate approaches in robotics (e.g. spatial dynamics [Mir96, Fea14, Sha09]), Kane's method, Udwadia-Kalaba equations, symbolic and recursive approaches and sometimes even writing and solving the equations directly on paper or using software like Mathematica or Maple.

As already stated, acceleration based methods are obtained from differentiating the constraint twice:

$$\ddot{\mathbf{c}}(\mathbf{x}) = \dot{\mathbf{J}}\mathbf{v} + \mathbf{J}\dot{\mathbf{v}} = \mathbf{0}. \tag{6.6}$$

In the case of one constraint the first term of (6.6) can be expressed as a quadratic form of the velocity: $\mathbf{v}^T\mathbf{H}\mathbf{v}$, where $\mathbf{H}$ is the Hessian matrix of the constraint [LR04]. In the general case $\mathbf{H}$ becomes a rank 3 tensor so it is easier to work with the matrix $\dot{\mathbf{J}} = \nabla\dot{\mathbf{c}}$ [Wit97].

We can manipulate equation (6.4) to obtain the term $\mathbf{J}\dot{\mathbf{v}} = \mathbf{J}\mathbf{M}^{-1}(\mathbf{f}_{ext} + \mathbf{J}^T\boldsymbol{\lambda})$ and substitute it in (6.6) to obtain a linear system:

$$\mathbf{A}\boldsymbol{\lambda} = -\mathbf{J}\mathbf{M}^{-1}\mathbf{f}_{ext} - \dot{\mathbf{J}}\mathbf{v}, \tag{6.7}$$

where $\mathbf{A} = \mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T$. Solving this system gives us an expression for the

Lagrange multipliers at any moment in time, given that we can compute the rest of the terms. This is important as the method is not dependent on the discretization chosen to solve the ODE in (6.4). One can choose the RK4 integrator for example (very popular in the 90s) and the constraint forces needed at each instant of time are computed using (6.7).

The main drawback is that given initial conditions that are not satisfying the constraint or numerical drift in the positions and velocities the method will not correct these errors. In order to address this problem we use Baumgarte stabilization:

$$\ddot{\mathbf{c}}(\mathbf{x}) + \alpha\dot{\mathbf{c}}(\mathbf{x}) + \beta\mathbf{c}(\mathbf{x}) = \mathbf{0}, \tag{6.8}$$

where α and β are tweakable parameters. For more information about choosing the parameters and their problems see [ACPR95]. For example one can choose $\alpha = 2\gamma$ and $\beta = \gamma^2$ where only γ has to be tweaked. Given the issues with Baumgarte stabilization several other post-stabilization and projection methods have been developed [CP03, AH04a, HLW06]. Still the acceleration based method was very popular in the 90s and together with a *linear complementarity problem* (LCP) formulation of contact and friction it was the main tool for rigid body simulation [Bar94].

6.2.2 Molecular dynamics

In computational molecular dynamics (MD) often very strong bonds are replaced by constraints. This is done in order to reduce the stiffness of the problem and speed up the computation. The algorithm SHAKE dates back to the 70s and it is based on a Verlet integrator and nonlinear equation solving [RCB77]. It was later extended to Velocity Verlet and phase space projection (i.e. the first derivative of constraint is also enforced): RATTLE [And83]. A lot of other variants exist, especially with the property of being symplectic [BKLS95, LR04]. These are all related in a way to the method of projection for solving DAEs [HLW06]:

$$\text{minimize } \|\delta\mathbf{x}\|_{\mathbf{M}}^2 \text{ subject to } \mathbf{c}(\mathbf{x}^{l+1}) = 0, \tag{6.9}$$

where $\delta\mathbf{x} = \mathbf{x}^{l+1} - \tilde{\mathbf{x}}$ is the difference between the new unknown position $\mathbf{x}$ and an initial candidate position $\tilde{\mathbf{x}}$ obtained through unconstrained integration. The $\mathbf{M}$-energy (or $\mathbf{M}$-metric L2 squared norm) $\|\mathbf{x}\|^2_{\mathbf{M}}$ is simply the quadratic form $\mathbf{x}^T\mathbf{M}\mathbf{x}$.

The SHAKE algorithm can be summarized as:

$$\mathbf{x}^{l+1} = 2\mathbf{x}^l - \mathbf{x}^{l-1} + h^2\mathbf{M}^{-1}(\mathbf{f}_{ext} + \mathbf{J}^T\boldsymbol{\lambda}), \tag{6.10}$$

$$\mathbf{0} = \mathbf{c}(\mathbf{x}^{l+1}). \tag{6.11}$$

This method works only at position level, but requires the storage of the previous position and a fixed time step. Here the candidate position is $\tilde{\mathbf{x}} = 2\mathbf{x}^l - \mathbf{x}^{l-1} + h^2\mathbf{M}^{-1}\mathbf{f}_{ext}$. Velocity can be approximated by a backward difference $\mathbf{v}^{l+1} = (\mathbf{x}^{l+1} - \mathbf{x}^l)/h$ or by a centered difference $\mathbf{v}^{l+1} = (\mathbf{x}^{l+1} - \mathbf{x}^{l-1})/2h$. For solving the nonlinear system any method can be used, but the name SHAKE often implies the Gauss-Seidel procedure presented in the original paper [RCB77].

The RATTLE scheme makes sure that both positions and velocities are projected on the phase space constraint manifold (making it more expensive):

$$\mathbf{v}^{l+1/2} = \mathbf{v}^l + \tfrac{h}{2}\mathbf{M}^{-1}(\mathbf{f}_{ext} + \mathbf{J}^T\boldsymbol{\lambda}), \tag{6.12}$$

$$\mathbf{x}^{l+1} = \mathbf{x}^l + h\mathbf{M}^{-1}\mathbf{v}^{l+1/2}, \tag{6.13}$$

$$\mathbf{0} = \mathbf{c}(\mathbf{x}^{l+1}), \tag{6.14}$$

$$\mathbf{v}^{l+1} = \mathbf{v}^{l+1/2} + \tfrac{h}{2}\mathbf{M}^{-1}(\mathbf{f}_{ext} + \mathbf{J}^T\boldsymbol{\mu}), \tag{6.15}$$

$$\mathbf{0} = \mathbf{J}\mathbf{v}^{l+1}. \tag{6.16}$$

Here $\boldsymbol{\mu}$ is a second set of Lagrange multipliers ensuring that the velocity level constraint (6.16) holds. Both SHAKE and RATTLE use the latest position candidate available (through unconstrained integration) to estimate the Jacobian before solving the constraint equation.

6.2.3 Computer graphics

In computer graphics constraints were introduced by Baraff and Witkin [Bar94, Wit97]. They are also responsible for a big part of the physical simulations at Pixar. Baraff's pioneering work on rigid body simulation with contact and friction [Bar97] lay at the foundation of physics engines. However his acceleration based method did not turn out to be the method of choice and it was superseded by the work of Stewart and Anitescu (more on this in the next chapter). Building from Stewart's method and using influences from molecular dynamics and DAE theory, Jakobsen introduced the first position based method in games [Jak01]. One could argue that *strain limiting* [Pro96, BFA02] dates back even earlier but we stress the fact that it was used only as a correction for explicit springs[5]. Later Müller et al. [MHHR07] further developed the method, switched to a Symplectic Euler integrator and gave it its now popular name: *position based dynamics* (PBD). Goldenthal made it more academic and presented it in a different form (decoupled from the Gauss-Seidel solver) called *fast projection* [GHF+07, Gol10]. Although not clear from his description, Stam seems to have used a similar approach in implementing the Nucleus physics engine in Autodesk Maya [Sta09, HS07].

Goldenthal gives a very comprehensive view on position based methods and explains why they need to be fully implicit (i.e. also in terms of constraint directions) in order to be stable and always have solution. This gives rise to a very large Newton system with $n + m$ unknowns [Gol10]. This was not the case for SHAKE as the constraint directions were considered constant. This approximation usually breaks down for structures that can have transverse oscillations like threads or membranes [TNGF15]. Goldenthal then introduces a faster iterative method in order to compensate for the complexity of the Newton system. This is nothing else than a *sequential quadratic programming* (SQP) [WN99] approach to solving the projection problem in (6.9). Under closer scrutiny the PBD method does the same thing but it only runs one Gauss-Seidel iteration to solve each sequential

[5] At most it can be considered as a hybrid approach, but this was not clear at that time and also we are concentrating in this thesis on pure constraint based methods.

linear system. The resulting solver is usually called *nonlinear Gauss-Seidel* (NGS) and its key difference to SHAKE is that it also updates the constraint directions[6].

In essence PBD is very intuitive: you first integrate the positions using external forces only (with Verlet or Symplectic Euler) and then correct each constraint in an iterative fashion by applying displacements. Note that you can either update velocities at the very end or during the solver by applying impulses (i.e. displacements divided by h). This seems deceptively simple but it hides a lot of its origin and derivation details that we outlined partly above. We tried to understand as much as possible where PBD really comes from and what it really is and came up with two equivalent answers:

- a position-based (index 3) DAE solver using projection or
- a nonlinear velocity time stepping (index 2) DAE solver with stabilization (more on this fixed point approach in the next chapter).

Given that PBD is intrinsically connected to the Gauss-Seidel relaxation scheme, we decided to denote together all position based methods as *position projection* methods or *nonlinear constrained dynamics* methods.

6.3 Velocity time stepping

Before going further and presenting our novel view on position based methods, we would like to introduce the velocity based method for comparison. This method is used mainly for contact and friction, but it can also handle bilateral constraints [TA11, Erl07]. It is obtained by discretizing the equations of motion (6.4) using a semi-implicit (or symplectic) Euler integrator[7] as in [ST96]. The velocity based method bears a lot of similarity to the *impulse based method* [Mir96, GBF03] and therefore Catto called his Gauss-Seidel solver Sequential Impulses [Cat05]. It is also strongly related to the acceleration formulation with the difference that the problem is solved after

[6]We have not found until this point any algorithm in MD that is similar to PBD.

[7]Explicit integrators cannot work as the constraint forces are not known, and implicit integrators actually give us position based methods.

the time discretization is performed resulting in an index 2 DAE in positions and velocities.

We will call these methods generically as *velocity time stepping* (VTS) methods because they are different from event-based simulators where the time step is divided in order to identify discontinuous events (VTS steps over them). Here is the most common formulation:

$$\mathbf{v}^{l+1} = \mathbf{v}^l + h\mathbf{M}^{-1}(\mathbf{f}_{ext} + \mathbf{J}^T\boldsymbol{\lambda}), \tag{6.17}$$

$$\mathbf{x}^{l+1} = \mathbf{x}^l + h\mathbf{v}^{l+1}, \tag{6.18}$$

$$\mathbf{0} = \mathbf{J}\mathbf{v}^{l+1} + \tfrac{\gamma}{h}\mathbf{c}(\mathbf{x}^l), \tag{6.19}$$

where γ is a Baumgarte-like stabilization parameter between 0 and 1 [TA11, Cat05]. Note that the Jacobian is computed at the beginning of the time step: $\mathbf{J} = \nabla\mathbf{c}(\mathbf{q}^l)$ and this is an important distinction from PBD and MD methods. The original formulation [ST96] had a different linearization of the constraints than the one in (6.19) and was later replaced by pure velocity constraints in [AP97]. The form in (6.19) was only introduced in [AH04a] as a form of constraint stabilization using one single LCP solve. The velocity time stepping scheme (VTS) can also be expressed in KKT matrix form[8]:

$$\begin{pmatrix} \mathbf{M} & -h\mathbf{J}^T \\ \mathbf{J} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{v}^{l+1} \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \mathbf{M}\mathbf{v}^l + h\mathbf{f}_{ext} \\ -\frac{\gamma}{h}\mathbf{c}(\mathbf{x}^l) \end{pmatrix}. \tag{6.20}$$

After taking the Schur complement (i.e. substituting $\mathbf{J}\mathbf{v}^{l+1}$) we obtain the following linear system:

$$h\mathbf{A}\boldsymbol{\lambda} + \mathbf{J}\tilde{\mathbf{v}} + \tfrac{\gamma}{h}\mathbf{c}(\mathbf{x}^l) = 0, \tag{6.21}$$

where $\mathbf{A} = \mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T$ and $\tilde{\mathbf{v}} = \mathbf{v}^l + h\mathbf{M}^{-1}\mathbf{f}_{ext}$.

[8] Term originating from the Karush-Kuhn-Tucker optimality conditions [WN99].

6.4 Nonlinear minimization

The fact that PBD uses projection was stressed from the very start by Jakobsen and he also makes reference to interior point methods. Although the nonlinear Gauss-Seidel solver is not an interior point method, this view did open to us a range of other possibilities for solving the position projection problem. NGS can be viewed as either an SQP approach or as a nonlinear gradient descent minimization method. In the former approach one can use any other solver of choice for the linear systems. Goldenthal for example used the PARDISO exact sparse solver [GHF+07].

We experimented with iterative solvers from the Conjugate Gradient (CG) family [PTVF07, She94]. Most of them worked given enough iterations, but there were stability issues when not using the Minimum Residual (MINRES) type of algorithms. We found out that the problem was not related to the system matrix but rather to the way the residual evolved along the iterations[9]. It turned out CG has a very erratic evolution of the residual as it focused on minimizing the error, whereas MINRES methods have a monotonic decrease in the residual as the aim is to minimize it at every step (see Figure 6.1). Our method of choice was in the end Conjugate Residuals (CR) [Saa03].

Our main contribution was eventually a direct tackle of the nonlinear minimization problem in (6.9) [FM14b, FM14a]. In order to be able to use a nonlinear gradient descent approach we had to convert the problem to its dual [WN99] which in the case of bilateral constraints is also unconstrained. This conversion is common practice in the constraint dynamics literature [MHNT15] as it is much easier to work with the dual variables, i.e. the Lagrange multipliers. There is still the possibility to solve the primal problem (6.9) using a different type of algorithm (like the interior point method) or to solve the saddle point problem in (6.20) or (6.25) but so far we have not seen any such competitive methods for interactive physics.

The implicit Euler integrator has also been shown to have such a mini-

[9]This makes sense because the residual equates constraint error in our case and this means extra potential energy added to the system resulting in additional kinetic energy in the next frames (e.g. the cloth coming alive from equilibrium).

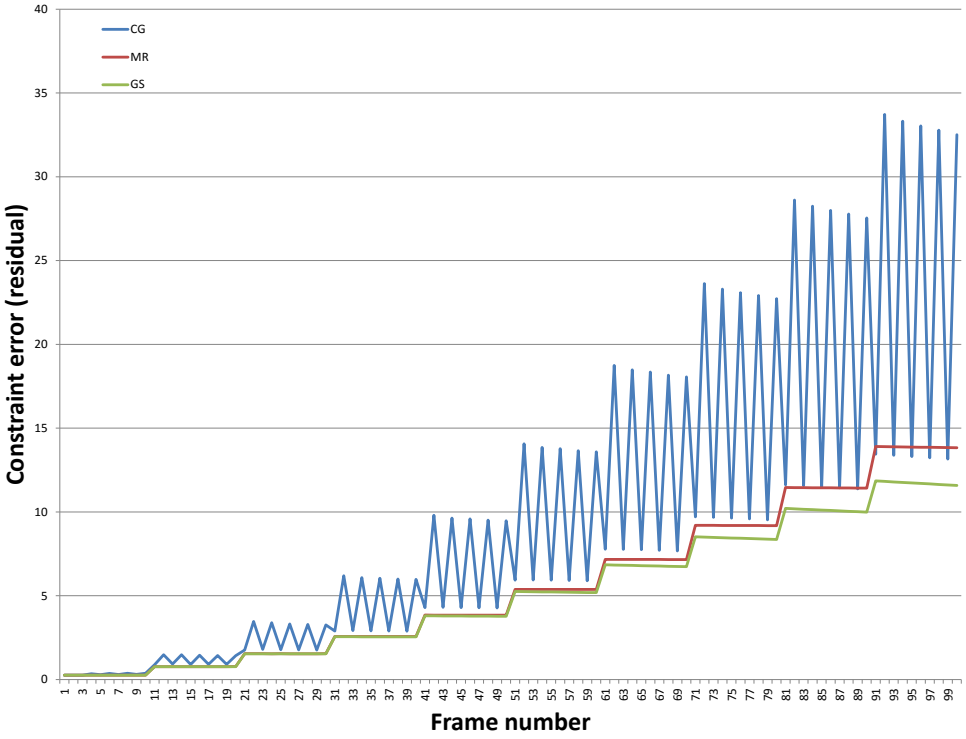


Figure 6.1: Plot of the residual using 3 different solvers.

mization structure in Section 4.3. The formulation in (4.6) with constraints added gives us an objective function close to the one used in projection:

$$\text{minimize } \tfrac{1}{2h^2}\Delta\mathbf{x}^T\mathbf{M}\Delta\mathbf{x} + V(\mathbf{x}^{l+1}) \text{ subject to } \mathbf{c}(\mathbf{x}^{l+1}) = 0, \qquad (6.22)$$

where $\Delta\mathbf{x} = \mathbf{x}^{l+1} - \mathbf{x}^l - h\mathbf{v}^l$. This is essentially the same formulation as in Projective Dynamics (PD) [BML+14]. It can be made the same as in (6.9) and in [Gol10] if we replace $\Delta\mathbf{x}$ by $\delta\mathbf{x}$ and eliminate the need for the external forces potential term. PD is different in that it incorporates the unconstrained integration inside the minimization and uses a different solving approach. But the main takeaway here is that projection is simply just the implicit Euler integration of constraint forces.

The question arises whether the minimization in (6.9) is related in any way to the variational principles of mechanics, which involve the stationary

point of the action functional. And the answer turns out to be positive for certain choices of integrators. It is remarkable to see how a general extremum problem can be turned into a minimization at every discretized time step (see Section 6.5 and [KYT+06b]).

The dual problem is obtained by computing the Lagrangian function $L(\delta\mathbf{x}, \boldsymbol{\lambda})$ of (6.9) and then minimizing it over both $\delta\mathbf{x}$ and $\boldsymbol{\lambda}$ [BV04]. As an intermediate step we obtain the following equivalent form to (6.9):

$$\text{minimize } L(\delta\mathbf{x}, \boldsymbol{\lambda}) = \tfrac{1}{2h^2}\delta\mathbf{x}^T\mathbf{M}\delta\mathbf{x} - \boldsymbol{\lambda}^T\mathbf{c}(\mathbf{x}^{l+1}). \tag{6.23}$$

Stating position projection as a dual nonlinear optimization problem is a premiere to our knowledge and we think it is a very general and versatile form:

$$\text{minimize } \tfrac{h^2}{2}\boldsymbol{\lambda}^T\mathbf{A}\boldsymbol{\lambda} + \boldsymbol{\lambda}^T\mathbf{c}(\mathbf{q}^{l+1}), \tag{6.24}$$

where $\mathbf{A} = \mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T$ (as for the other methods). Note that the constraint Jacobian is considered here at the end of the time step in an implicit manner: $\mathbf{J} = \nabla\mathbf{c}(\mathbf{q}^{l+1})$.

SQP applied to the primal problem (6.9) gives us the following KKT system at each iteration (or the full Newton method):

$$\begin{bmatrix} \mathbf{M} - h^2\mathbf{H}_k\boldsymbol{\lambda}_k & -h^2\mathbf{J}_k^T \\ \mathbf{J}_k & \mathbf{0} \end{bmatrix} \begin{pmatrix} \delta\mathbf{x}_{k+1} \\ \delta\boldsymbol{\lambda}_{k+1} \end{pmatrix} = \begin{pmatrix} \mathbf{M}\delta\mathbf{x}_k - h^2\mathbf{J}_k^T\boldsymbol{\lambda}_k \\ -\mathbf{c}(\mathbf{x}_k) \end{pmatrix}. \tag{6.25}$$

Usually we omit the second derivative term [Gol10] and obtain a matrix very similar to the one in (6.20) (see a discussion on this omission later in Section 6.10). Then we can take a Schur complement and express the problem only in terms of the dual variables:

$$\begin{aligned} \mathbf{0} = \mathbf{c}(\mathbf{x}_k) + h^2\mathbf{J}_k\mathbf{M}^{-1}\mathbf{J}_k^T\delta\boldsymbol{\lambda}_{k+1} = \mathbf{A}_k\boldsymbol{\delta}\lambda_{k+1} + \mathbf{b}_k, \\ \mathbf{x}_{k+1} = \mathbf{x}_k + \delta\mathbf{x}_{k+1}, \end{aligned} \tag{6.26}$$

where $\delta\mathbf{x}_{k+1} = h^2\mathbf{M}^{-1}\mathbf{J}_k^T\delta\boldsymbol{\lambda}_{k+1}$ as obtained from (6.25).

This is also equivalent to the following QP at every iteration:

$$\text{minimize } \tfrac{h^2}{2}\delta\boldsymbol{\lambda}^T\mathbf{A}_k\delta\boldsymbol{\lambda} + \delta\boldsymbol{\lambda}^T\mathbf{b}_k. \tag{6.27}$$

Velocity based methods can be reduced to a similar minimization form as in (6.9) [TA11] and the resulting dual problem is actually a *quadratic program* (QP) and thus easier to solve. We argue that in fact it is just a linearization of PBD and that both methods are based on projection. We will give a more complete argument in Section 7.5 after adding contact and friction, but it should not be to hard to see even at this point after comparing the primal formulation in [Ani06] with (6.9) or the dual one in [MHNT15] with (6.24). Or just notice the similarity between equations (6.21) and (6.26).

6.5 Variational minimization structure

Actually we will try to make the case in what follows that most DAE solvers can be recast as minimization problems. The reasoning is as following:

1. all dynamics is based on Hamilton's principle of stationary action;
2. the index 3 DAE equations of motion are the same as extremizing the Lagrangian with constraints (variational optimality conditions);
3. through the tools of discrete mechanics Hamilton's principle can be turned into a minimization at every time step [KYT+06a];
4. the optimization form of implicit symplectic integrators (e.g. Newmark [WKMO99]) with constraints is equivalent to projection[10];
5. the Newmark integrator has the same optimization formulation as implicit Euler for $\beta = 1$;
6. implicit Euler integration with position constraints at the end of the time step is in fact PBD or fast projection;

[10]With an extra explicit potential term in the objective; the case of mixing implicit integration of stiff external forces with constrained dynamics is not considered here.

7. velocity time stepping is a linearized version of position projection and so in turn also a minimization problem.

To summarize, projection is a discretized way of expressing Hamilton's principle and expresses as a constrained minimization prolem both position and velocity based methods for solving DAEs. Acceleration based methods can also be expressed as a minimization through Gauss' principle and can be regarded as enforcing the second layer of hidden constraint besides velocities in the index 3 DAE. In a nutshell, all the methods can be expressed as a minimization which in turn can be recast to a projection on the constraint manifold or its tangent bundles.

We will follow the derivation in [KMOW99] to show how the Newmark integrator can be recast as a minimization problem. We chose this integration scheme because it is proven to be variational and symplectic (under certain circumstances and a proper choice of parameters) and it is quite general. Also for non-zero β it is implicit and this is required for projection to work[11]. In general, not all integrators (be they variational) can be reformulated as a minimization problem unless they satisfy a *variational integrability property* [KYT+06b, Lew03]. We will not delve into the details here as they are quite cumbersome, but fortunately most of the integrators satisfy this condition.

If we consider external forces as explicit and non-dissipative and also add constraints we get the following Newmark discretization:

$$\mathbf{v}^{l+1} = \mathbf{v}^l + h[(1-\gamma)\mathbf{a}^l + \gamma\mathbf{a}^{l+1}], \tag{6.28}$$

$$\mathbf{x}^{l+1} = \mathbf{x}^l + h\mathbf{v}^l + \tfrac{h^2}{2}[(1-2\beta)\mathbf{a}^l + 2\beta\mathbf{a}^{l+1}], \tag{6.29}$$

$$\mathbf{M}\mathbf{a}^{l+1} = -\nabla V(\mathbf{x}^l) + \nabla\mathbf{c}(\mathbf{x}^{l+1})\boldsymbol{\lambda}. \tag{6.30}$$

This can be rewritten as:

$$\tfrac{1}{h^2}\mathbf{M}(\mathbf{x}^{l+1} - \tilde{\mathbf{x}}) = -\beta\nabla V(\mathbf{x}^l) + \beta\nabla\mathbf{c}(\mathbf{x}^{l+1})\boldsymbol{\lambda}, \tag{6.31}$$

[11] It also works for semi-implicit Euler and Verlet but we tackle the fully implicit case.

where $\tilde{\mathbf{x}} = \mathbf{x}^l + h\mathbf{v}^l + \frac{h^2}{2}(1-2\beta)\mathbf{a}^l$. It becomes clear now how this can be recast as the minimization problem:

$$\text{min. } \frac{1}{2h^2}(\mathbf{x}^{l+1} - \tilde{\mathbf{x}})^T\mathbf{M}(\mathbf{x}^{l+1} - \tilde{\mathbf{x}}) + \beta V(\mathbf{x}^l) - \beta\boldsymbol{\lambda}^T\mathbf{c}(\mathbf{x}^{l+1}). \tag{6.32}$$

For $\beta = 1$ this becomes the same objective as for projection (see equation (6.23)) and so we can view (6.32) as a generalization of projection. One can use the velocity update equation (6.28) to implement a Newmark based position projection scheme. In conclusion we have shown that the method of constraint projection as a minimization process is in fact deeply rooted in the discretization of the variational principle of stationary action (steps 3 to 6 above).

6.6 Solvers

In this section we take a closer look at iterative solvers both for linear systems and nonlinear optimization problems. In our experience they are easier to understand in the former case and then can be extended to optimization - the very idea of gradient descent and linear CG comes from minimizing a convex quadratic objective. In fact most nonlinear equations can be better handled numerically in their equivalent optimization form as there exist more algorithms and they are better conditioned. The only requirement is that the problem is convex [BV04] and this is true in most cases presented in this chapter (frictional contact is an exception presented in the next chapter).

The two important classes of methods analyzed are relaxation solvers and Krylov subspace methods, better known as the conjugate gradient family of algorithms. At a closer scrutiny the two are quite similar as they both are line searches:

$$\boldsymbol{\lambda}_{k+1} = \boldsymbol{\lambda}_k + \alpha\mathbf{d}_k, \tag{6.33}$$

and at the same time gradient descent methods as the direction $\mathbf{d}$ is usually the gradient or something related [RA05]. Gauss-Seidel, SOR and Jacobi are shown in [BETC14] to be coordinate descent methods [LY84]. Steepest Descent (SD) [She94], MINRES and Nesterov's method [MHNT15] are

all gradient search methods and they differ in the way the step size α is computed. The ideas of conjugacy and Krylov subspace are used to choose better search directions and improve convergence. These conjugate gradients or residuals methods are also usually accompanied by pre-conditioning but we did not spend time on this avenue for this thesis [She94, Saa03]. Similar descriptions of iterative solvers to the one in this section can also be found in two of our published articles [FM14b, FM14a].

6.6.1 Relaxation

Relaxation solvers are simple iterative methods for solving large sparse linear systems. They are also called *stationary* as the recursion formula does not change or *splitting* methods as they rely on splitting the system matrix in 2 or 3 components. They are also called *preconditioners* when used to improve the convergence of Krylov methods or *smoothers* in the context of multigrid [Str07]. They include the Jacobi, Gauss-Seidel and Successive Over-Relaxation (SOR) methods.

Our aim is to solve the linear system:

$$\mathbf{A}\boldsymbol{\lambda} + \mathbf{b} = 0. \tag{6.34}$$

The Jacobi method is based on the splitting: $\mathbf{A} = \mathbf{D} + \mathbf{E}$, where $\mathbf{D}$ is a the diagonal of $\mathbf{A}$. We obtain thus the following recurrence:

$$\boldsymbol{\lambda}_{k+1} = \mathbf{D}^{-1}(-\mathbf{E}\boldsymbol{\lambda}_k - \mathbf{b}), \tag{6.35}$$

where the superscript k indicates here the iteration number.

We introduce now the very important notion of the residual:

$$\mathbf{r}_k = -(\mathbf{A}\boldsymbol{\lambda}_k + \mathbf{b}). \tag{6.36}$$

It is defined at each iteration and it signifies the error in solving the system (6.34) in contrast to the actual solution error. In the optimization view (see (6.38) below or [BETC14]) the residual corresponds to the gradient of

the objective function f and this formula holds also for the nonlinear case: $\mathbf{r} = -\nabla f$ [She94]. In the context of constrained dynamics the residual has an even more precise meaning: the constraint error, i.e. the relative velocity plus stabilization for VTS [TBV12] or the positional constraint error for position projection [FM14b].

Using the residual we can rewrite (6.35) as:

$$\boldsymbol{\lambda}_{k+1} = \boldsymbol{\lambda}_k - \omega \mathbf{D}^{-1} \mathbf{r}_k. \tag{6.37}$$

Here we introduced the (under-)relaxation factor $\omega < 1$ in order to guarantee the convergence of the method. The convergence rate depends on the spectral radius of the growth matrix $-\mathbf{D}^{-1}\mathbf{E}$ and usually we need to scale it down as its modulus is greater than one [She94]. Bridson proposes a method of computing ω as the inverse of the number of constraints incident to each particle [BFA02].

The Gauss-Seidel method is similar and uses a different splitting: $\mathbf{A} = \mathbf{D} + \mathbf{L} + \mathbf{U}$. The recurrence formula is easier to see as a per component update:

$$\lambda_i^{k+1} = \frac{\omega}{A_{ii}} \left(-b_i - \sum_{j=1}^{j<i} A_{ij} \lambda_j^{k+1} - \sum_{j=i+1}^{m} A_{ij} \lambda_j^k \right),$$

where ω is an over-relaxation term this time, slightly greater than one, used in SOR. The main advantage of Gauss-Seidel is that it uses the most recent information (propagation is faster) and has a better convergence rate than Jacobi; SOR converges even faster [Saa03].

All relaxation methods basically perform a local solve of each equation, i.e. solve for each unknown at a time by using current estimates of the other unknowns. This is why they are often called *local solvers*. Like any iterative method relaxation needs to start from an initial guess of the solution and here one possible optimization is *warm starting* [Cat05]. We have not focused much on warm starting as we wanted to compare unaltered implementations of the solvers and also we had issues with bilateral constraints where constraint forces can change quite a lot from frame to frame. Still, if suitable to use, we consider this optimization technique indispensable in real world

implementations.

Given that the convergence rate depends ultimately on the largest eigenvalue of the system matrix, this means they reduce the error only slightly for eigenvectors corresponding to the largest eigenvalues. This usually corresponds to high frequency components of the mesh used to spatially discretize the problem, e.g. $\mathbf{A}$ can correspond to the discrete Laplacian on a grid or mesh. Thus relaxation methods are only efficient on low frequencies (low-pass filter) and this is why they are called smoothers [Str07]. This problem can be alleviated by running relaxation on coarser grids and this is the strategy of multigrid solvers - an implementation in the context of PBD can be found in [Mül08] and one for VTS in [OR07]. We have not studied multigrid methods during our doctoral research but we believe they are a good path to pursue in the future, especially by mixing Krylov methods with relaxation.

6.6.2 Krylov subspace methods

These methods include the well known Conjugate Gradient (CG) algorithm and its variants. We will only describe it here shortly, while a comprehensive introduction can be found in [She94]. This family of methods tries to solve the linear system in (6.34) by casting it to a quadratic minimization problem:

$$\text{min. } f(\boldsymbol{\lambda}) = \tfrac{1}{2}\boldsymbol{\lambda}^T\mathbf{A}\boldsymbol{\lambda} + \mathbf{b}^T\boldsymbol{\lambda}. \tag{6.38}$$

This only works if the matrix $\mathbf{A}$ is symmetric and positive-definite (the objective function is convex). The simplest approach is to start from an initial guess and descend down the negative gradient of the function f, which is precisely the residual vector $\mathbf{r}$ defined in the previous section. This method is called Steepest Descent (SD) but it does not have very good convergence, so CG appeared as an improvement based on conjugacy (i.e. $\mathbf{A}$-orthogonality). You can find more details about the theory behind it, the algorithm itself, why it's called a Krylov subspace method, preconditioning and other variants for more general matrices in [She94] and (a more advanced text) [Saa03].

We will now mention briefly the *minimum residual* methods that were initially developed for indefinite symmetric matrices. These methods try to

minimize the quadratic norm of the residual $\|\mathbf{r}\|^2$ instead of the function f [PTVF07]. The methods described above can be easily converted to their minimum residual variant by replacing all dot products of the form $\mathbf{a} \cdot \mathbf{b}$ with $\mathbf{a}^T\mathbf{A}\mathbf{b}$ (the $\mathbf{A}$ matrix dot product).

The minimum residual counterpart of CG is the Conjugate Residuals (CR) algorithm. This was further generalized by Saad into the GMRES method [Saa03]. This is of importance because we found that the CR algorithm is more stable than CG when applied to constraint projection and for small number of iterations [FM14b], although there were reports of successfully using CG too [HCJ+05]. Another application of this type of methods was in the context of contact mechanics where a projected optimization variant GPMINRES was used to solve a *cone complementarity problem* (CCP) [HATN12].

6.6.3 Accelerated Jacobi

We now turn our attention to the nonlinear optimization formulation in (6.24) for position projection in order to present our contributions. Extending relaxation to the nonlinear case is done by updating the system matrix at every iteration - remember NGS [OR70]. We have already described nonlinear CG [She94] in Chapter 3.

For applying MINRES to the nonlinear case there are two equivalent ways of viewing the process: either as a one step MINRES-Newton scheme for solving (6.26) or as a gradient descent method for minimizing (6.24). If we take the residual to be $\mathbf{r} = -\nabla\mathbf{c}(\mathbf{x})$ at the beginning of each iteration and use the fact that $\mathbf{A} = \mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T$ we get a new method for solving position based constraints, which is presented in Algorithm 2.

This solver has slightly better convergence than Jacobi while at the same time being more stable (without any modifications as needed for Jacobi). The algorithm computes constraint errors and then tries to use them to move the particles along the constraint direction in the hope of minimizing the resulting error. The trick is not to use the whole error amount, but a fraction α of it (Jacobi does the same thing but uses the inverses of the

Algorithm 2 Nonlinear Minimum Residual

$\mathbf{x} = \tilde{\mathbf{x}}$
for `iter` $= 1$ **to** `numIters` **do**
  $\mathbf{r} = -\mathbf{c}(\mathbf{x})$
  $\mathbf{J} = \nabla\mathbf{c}(\mathbf{x})^T$
  $\mathbf{y} = \mathbf{J}^T\mathbf{r}$
  $\mathbf{p} = \mathbf{A}\mathbf{r} = \mathbf{J}\mathbf{M}^{-1}\mathbf{y}$
  $\alpha = \frac{\mathbf{r}^T\mathbf{p}}{\mathbf{p}^T\mathbf{p}}$
  $\mathbf{x} = \mathbf{x} + \alpha\mathbf{M}^{-1}\mathbf{J}^T\mathbf{r}$
end for

diagonal elements of $\mathbf{A}$). In our case this fraction needs to be less than the reciprocal of the spectral radius of $\mathbf{A}$ and also to vary smoothly.

Another way of seeing this procedure is that at every iteration a low stiffness elastic force is being explicitly integrated for every constraint. The stiffness of the spring is $\kappa = \alpha/h^2$ while the resulting stiffness is proportional to the square of the number of iterations. By analyzing the stability of the integration growth matrix one can see why $\alpha < 1/\rho(\mathbf{A})$ (the norm of the eigenvalues has to be less than one). In this way we can robustly treat stiff systems whose natural period is less than the time step h by iteratively applying a "safe stiffness" elastic force.

As an optimization, if one can estimate $\rho(\mathbf{A})$ in advance, we can set α to its reciprocal and we cut down on a lot of calculations. This is actually very similar to a modified version of Jacobi where the displacements are scaled down by $\omega \leq 2/\rho(\mathbf{A})$ [TBV12]. An easy way to compute the spectral radius of a matrix is through the power method: an iterative method very similar to SD or MINRES that outputs the eigenvector corresponding to the largest eigenvalue in absolute value. Once we have a good approximation of this eigenvector we can easily calculate the spectral radius and also update it very quickly every frame to account for the changes in $\mathbf{A}$. This optimization works well in practice and is very stable but has slightly poorer convergence.

The CR method (Algorithm 3) is very similar to MINRES but this time the search directions $\mathbf{d}$ are chosen in a special way. In CG they are conjugate or $\mathbf{A}$-orthogonal, i.e. $\mathbf{d}^T\mathbf{A}\mathbf{d} = 0$. In CR we want the residuals to be conju-

Algorithm 3 Nonlinear Conjugate Residuals

$\mathbf{x} = \tilde{\mathbf{x}}$
for `iter` $= 1$ **to** `numIters` **do**
  $\mathbf{r} = -\mathbf{c}(\mathbf{x})$
  $\mathbf{J} = \nabla\mathbf{c}(\mathbf{x})$
  $\mathbf{y} = \mathbf{J}^T\mathbf{r}$
  $\delta_{new} = \mathbf{y}^T\mathbf{M}^{-1}\mathbf{y}$
  if `iter` $== 0$ **then**
    $\beta = 0$
  else
    $\beta = \frac{\delta_{new}}{\delta_{old}}$
  end if
  $\mathbf{d} = \mathbf{r} + \beta\mathbf{d}$
  $\delta_{old} = \delta_{new}$
  $\mathbf{y} = \mathbf{J}^T\mathbf{d}$
  $\mathbf{p} = \mathbf{J}\mathbf{M}^{-1}\mathbf{y}$
  $\alpha = \frac{\mathbf{r}^T\mathbf{p}}{\mathbf{p}^T\mathbf{p}}$
  $\mathbf{x} = \mathbf{x} + \alpha\mathbf{M}^{-1}\mathbf{J}^T\mathbf{d}$
end for

gate, which in turn makes the search directions to be $\mathbf{A}^2$-orthogonal [Saa03]. This speeds up convergence while at the same time keeping the simulation stable even for small number of iterations.

The algorithm does have problems dealing with large time steps. In order to stabilize the simulation we resorted to clamping down β below 1 and α below a maximum value, e.g. $1/\rho(\mathbf{A})$ or even larger. Still we found the algorithm to have stability issues around equilibrium as it does not dissipate all the energy, but the oscillations have very small amplitude and are hard to detect.

We found experimentally that β grows very rapidly from 0 to 1 and then stays almost constant. Also clamping it below 1 improved stability (see Figure 6.2). This is not surprising given that the residual should always decrease at every iteration. Given that the choice of β is not unique [HZ06],

we chose to approximate the calculation of β with:

$$\beta_k = \min\left(a\left(\frac{k}{k_{max}}\right)^b, 1\right), \tag{6.39}$$

where $a \geq 1$ and $b < 1$ and k_{max} is the maximum number of iterations. We obtained very good results and an even more stable simulation. In order to increase convergence one can make β grow steeper by lowering b and increasing a, but this may introduce jitter. We found $a = 1$ and $b = 0.6$ to be good values. Both CR and the power function optimized version have very good convergence, similar to Gauss-Seidel or even better in many cases (see Figure 6.3).

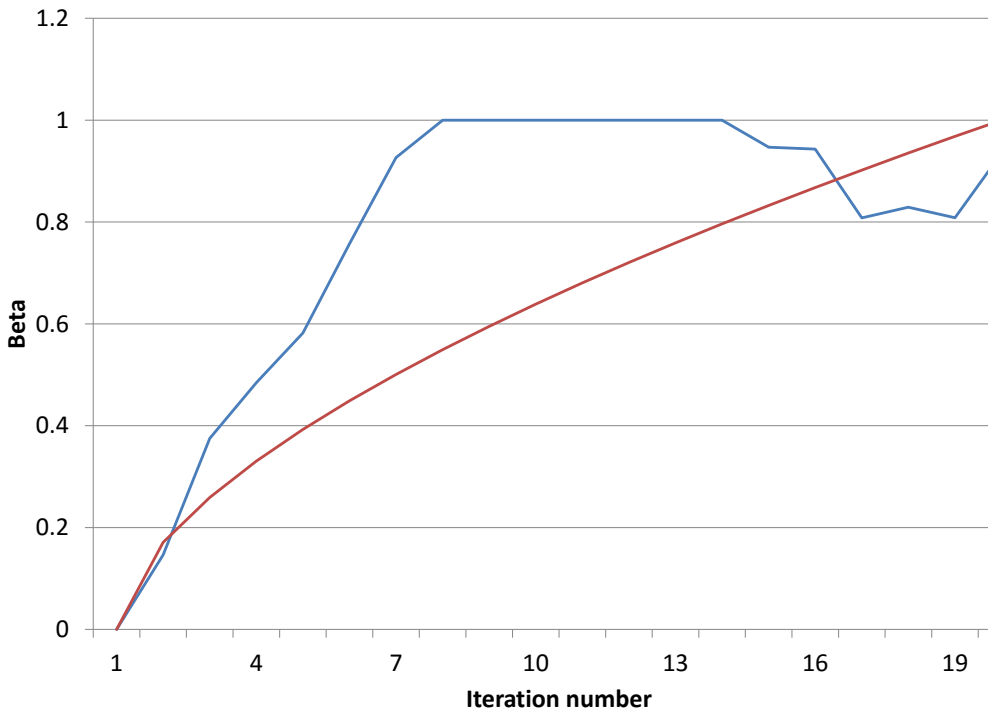


Figure 6.2: Plot of β over 20 iterations of the CR solver: original formula clamped below 1 (blue) and power function approximation (red).

If we set α fixed in the CR method just like we did for MINRES we can

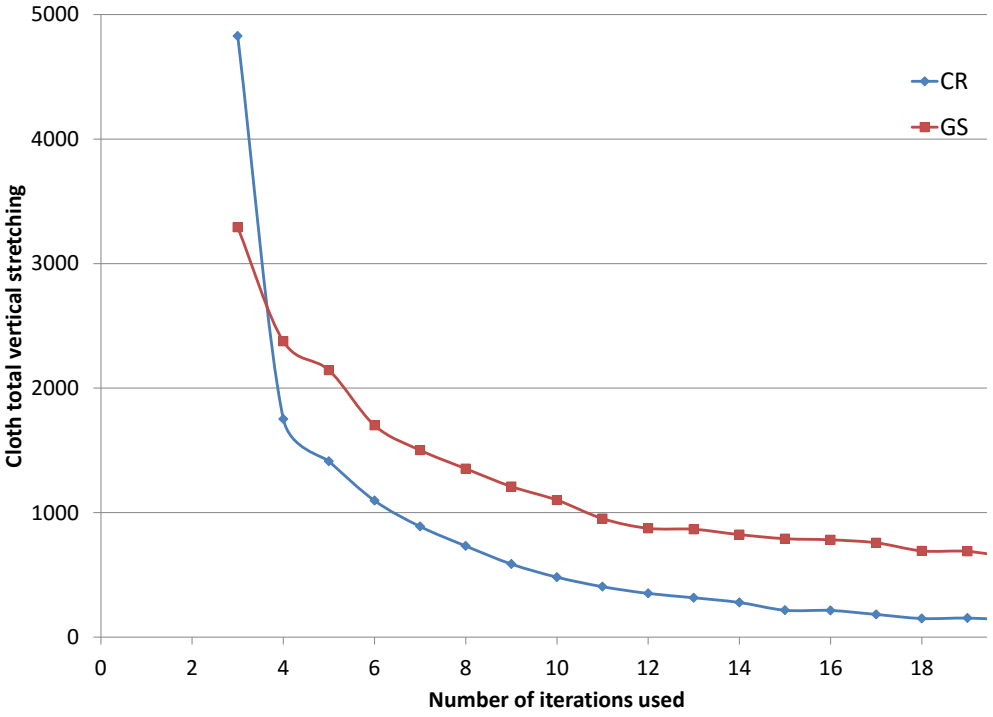


Figure 6.3: Plot of the total stretching at equilibrium of a 50×50 piece of cloth relative to the number of iterations for GS (red) and optimized CR (blue).

make the following simplification by using the descent direction from the previous iteration:

$$\delta\boldsymbol{\lambda}_{k+1} = -\alpha\mathbf{d}_k = -\alpha(\mathbf{r}_k + \beta_k\mathbf{d}_{k-1}) = -\alpha\mathbf{r}_k - \beta_k\delta\boldsymbol{\lambda}_k. \tag{6.40}$$

This way we don't need to store the descent direction any more and we get a simpler version of CR. We still have to provide a value for α and we could use $\alpha_0 = 1/\rho(\mathbf{A})$, but we can make things even simpler by using the step sizes from (6.35) and obtain a version of Jacobi with increased convergence:

$$\boldsymbol{\lambda}_{k+1} = \boldsymbol{\lambda}_k - \frac{\omega}{A_{ii}}\mathbf{r}_k - \beta_k\delta\boldsymbol{\lambda}_k, \tag{6.41}$$

where $\omega \leq 1$. We tested this improvement on GS too, but we had to lower

the value of ω a lot in order to stabilize the simulation. Also our focus is on the Jacobi method as it is easier to parallelize.

After looking more closely at the accelerated projected gradient descent (APGD) or Nesterov's method in [MHNT15] we came to the conclusion that it can be reduced to the same form as in (6.41). They both have the same momentum term added at the end with the difference that both α and β are computed differently. We decided to use the same step size as in Jacobi and adapt the following formulas from [MHNT15]:

$$\theta_{k+1} = \tfrac{1}{2}(-\theta_k^2 + \theta_k\sqrt{\theta_k^2+4}), \theta_0 = 1 \tag{6.42}$$

$$\beta_{k+1} = \frac{\theta_k(1-\theta_k)}{\theta_k^2+\theta_{k+1}}. \tag{6.43}$$

We called the resulting scheme "accelerated Jacobi". It worked very well in practice despite not having a rigorous mathematical derivation. We used it mainly for our parallel implementations, but one can still use CR or APGD instead if a more established method is desired.

6.7 Applications

As described in the previous chapter there are many applications of constrained dynamics. These involve mainly particle systems where very stiff springs are replaced by rigid links, but also more complex constraints like joints for rigid bodies, dihedral angle for cloth bending, contact and friction, volume and area constraints and even strain and finite elements.

The simplest didactic examples are a 1D spring or pendulum. The latter can be extended to a double pendulum and further to a kinematic chain. The beauty of constrained based dynamics is that it also allows for loops in the chain. In practice, these connected particle system are used for the simulation of threads, hair, cloth and soft bodies.

Although there have been reports of cloth simulation using VTS [Lac07, HCJ+05] most of the constrained based simulation of cloth in the past decade was done using PBD or some kind of position projection [MHHR07, GHF+07,

BBD09, EB08]. This is because PBD is much more stable than VTS, being fully nonlinear and implicit. Soft bodies were also simulated with PBD using a variety of techniques [BMOT13]. FEM was also touched upon using VTS [SLM06] and PBD [BKCW14, BML+14] (strain papers too). Therefore position projection methods have proven to be a very versatile tool in simulation and that is also the reason we set out to improve it both in theory and in practice. Besides developing a new parallel iterative solver that you read about in Section 6.6.3 we also derived and implemented the first accurate position-based FEM solver using constrains (see Section 6.11).

6.8 Regularization

In this section we present one of our most important theoretical contributions: that constrained based dynamics using nonlinear and implicit position projection with regularization is equivalent to the implicit Euler integration of elastic forces.

Regularization is a technique used to make optimization problems more tractable numerically [PPR11]: it makes the diagonal of the system matrix in (6.20) non-zero and ensures existence of solutions. This was used for solving rigid body constraints and even Lagrangian fluids [MMCK14]. It is a widely used technique in many fields for solving ill-posed problems, least squares problems (e.g. Tikhonov regularization), bi-criterion optimizations and saddle point problems [BV04, Str07].

A similar regularization used in rigid body engines also known as *constraint force mixing* [Smi06] or *soft constraints* [Cat10] was also compared to implicitly integrating stiff springs. This approach can be shown to be physical [Lac07] and can be generalized to continuous elastic materials as was done in [SLM06]. Even if not trying to simulate real elasticity, regularization can be a great tool for controlling how the positions and velocities are projected on the constraint manifold (as sometimes a too aggressive position correction can generate too high velocities and violent phenomena).

Regularization can be expressed as adding a small feedback term to the

constraint equation:

$$\tilde{\mathbf{c}}(\mathbf{x}) = \mathbf{c}(\mathbf{x}) + \epsilon\boldsymbol{\lambda} = \mathbf{0}. \tag{6.44}$$

This gives us $\boldsymbol{\lambda} = -\epsilon^{-1}\mathbf{c}(\mathbf{x})$ that we can substitute in (6.2) and obtain a force $\mathbf{f} = -\epsilon^{-1}\mathbf{c}(\mathbf{x})\nabla\mathbf{c}(\mathbf{x})$. If we denote by $\kappa = 1/\epsilon$ then we can see that this is equivalent to having a potential energy of the form $\frac{\kappa}{2}\|\mathbf{c}(\mathbf{x})\|^2$ which is precisely the elastic spring energy with the stiffness κ [Lac07].

In [FM15a] we showed that implicit Euler integration of elastic media is equivalent to regularized projection. For this we used the minimization formulation in (4.6). The potential term of the constraints V_c can be either a quadratic elastic energy or the Lagrange multiplier potential energy of the constraints introduced in Section 6.1 (provided the constraints are regularized). So the difference between implicit integration and constraint projection is the same as choosing between penalty and Lagrange multiplier methods in numerical optimization. We rewrite equation (4.6) to better stress our result:

$$\text{minimize } \tfrac{1}{2h^2}\Delta\mathbf{x}^T\mathbf{M}\Delta\mathbf{x} + V_c(\mathbf{x}) + V_{ext}(\mathbf{x}^l), \tag{6.45}$$

where V_c can be either $\frac{\kappa}{2}\|\mathbf{c}(\mathbf{x})\|^2$ or $-\boldsymbol{\lambda}^T\tilde{\mathbf{c}}(\mathbf{x})$. This means that there is no difference between the two methods and they are equivalent. The only requirement is that the constraints are regularized as in (6.44). So in order to achieve the same results using constrained dynamics as with implicit integration we need three ingredients: nonlinearity, implicit constraint directions and regularization.

We are clearly not the first to add compliance to constraints in order to soften them. Also, the analogy to implicit integration of springs was done before, but from a VTS perspective that may not have been correct[12]. The regularization technique in 6.44 was indeed shown to be the most physical

[12]This analogy identifies two parameters for regularization and stabilization from stiffness and damping coefficients. But the regularization term represents compliance on its own, so damping should come from somewhere else. We argue that the sources include the dissipativity of the integration scheme, the inelasticity of contacts and an explicit damping model, but not Baumgarte stabilization. This is why we turn our attention to a full implicit constraint solver for making the analogy. We also restrict ourselves to adding compliance only and leave damping for later.

one. The idea that constraints are the limit of infinitely stiff forces was also touched by other authors [AP02, Erl05, EB08]. Arnold gives the same argument in the context of Lagrangian mechanics on manifolds [Arn13] and Lacoursière cites the theorem of Rubin and Ungar [Lac07].

But we have not found any clear explanation to why PBD simulations of cloth look so similar to the ones done by linearly implicit integration. This is why we think we are the first to take the argumentation to the end and prove that the penalty method is just another way of solving the constrained dynamics problem in (6.9). And given that stiffness is a physical parameter it makes more sense to regularize constraints, rather than holding on to the ideal case. Moreover, in the case of numerical implementations we will never be able to satisfy the constraints exactly or we do not afford the computational expense. The difference from zero in (6.44) is the $\epsilon\boldsymbol{\lambda}$ feedback term which gives us precisely the amount of compliance added to the system. This is the explanation to why fewer iterations of NGS make the cloth springier.

Regularization amounts to replacing the lower right block of the KKT matrix in (6.20) or (6.25) with the compliance matrix $\mathbf{C}^{-1}$ which can be defined as $\epsilon\mathbf{1}$ like above or as the inverse of a more general stiffness matrix $\mathbf{C}$. Note that the matrix $\mathbf{C}$ is predominantly diagonal and not the same as the tangential stiffness matrix $\mathbf{K} = \nabla\mathbf{f} = -\nabla^2 V_c$ (Section 4.3). They are usually related through $\mathbf{K} = \mathbf{J}^T\mathbf{C}\mathbf{J}$ and the elastic potential is given by:

$$V_c(\mathbf{x}) = \tfrac{1}{2}\mathbf{c}(\mathbf{x})^T\mathbf{C}\mathbf{c}(\mathbf{x}). \tag{6.46}$$

Using regularization by $\mathbf{C}$ the linear system in (6.26) transforms to:

$$(h^2\mathbf{A}_k + \mathbf{C}^{-1})\delta\boldsymbol{\lambda}_{k+1} + \mathbf{b}_k = 0. \tag{6.47}$$

The advantage of this formulation is that, while equivalent to implicit integration, it is better numerically conditioned: as the stiffness increases, the compliance term vanishes and the system matrix tends to a constant (at least at frame or iteration level) [SLM06]. This is not the case for the modified mass matrix in (4.9) which will grow to infinity.

Looking more closely at the equation (6.47) we notice that it is not that different from (6.26). We can regard the change to the diagonal of the system matrix as a scaling through a relaxation factor as in [Jak01]. As noted in [MHHR07] this factor is highly nonlinear and we are now able to express this exact nonlinear relationship to the linear spring stiffness value:

$$\omega_j = (1 + (h^2 \kappa_j A_{jj})^{-1})^{-1} < 1, \tag{6.48}$$

where κ_j is the stiffness of spring j. On the other hand, if we use $\omega > 1$ we obtain SOR which may converge faster and produce better inextensibility, i.e. stiffer cloth for less iterations.

Another result we obtained from this equivalence is an explanation to why CG works so well in implicit integration but not so much for constraints where CR behaves better [FM15a]. We base our result on the fact that implicit methods work in velocity (or displacement) space and projection methods work in constraint space. These are nothing else than primal and dual variables of the same regularized optimization problem (6.45). The transformation from velocity space to constraint space at every point is given by the matrix $\mathbf{T} = \frac{1}{h\kappa}\mathbf{A}^{-1}\mathbf{J}$ and the reciprocal by $\mathbf{Q} = h\kappa\mathbf{M}^{-1}\mathbf{J}^T$. Thus, in order to run CG in constraint space we need to convert the residual in that space. In the end we obtained a new update formula for β which resembles the one in CR (see Section 7 in [FM15a]). But more investigation should be done here in order to identify the best Krylov method suited for constrained dynamics.

6.9 Energy dissipation and damping

We have already mentioned that for preserving energy during the simulation it is best to use symplectic integrators (Section 4.2). For DAEs an option is to use symmetric projection [HLW06], i.e. a method that can be reversed in time. Goldenthal proposes an alternative symmetric Velocity Verlet scheme [Gol10]. Another option is to use Implicit Midpoint that promises full energy conservation, but in practice it can become unstable. This is why in [FM15a]

we propose a projection scheme based on the Newmark integrator that can be tuned (see also Section 6.5). A similar projection method was developed in [EB08] where they use the BDF-2 scheme as it dissipates less energy than Backward Euler.

Another of our contributions is to add damping in a physical and credible manner to the position projection formulation. In general we can do this by using a Rayleigh dissipation function [Lac07]:

$$\varphi(\mathbf{v}) = \tfrac{1}{2}\dot{\mathbf{c}}(\mathbf{q})\mathbf{R}\dot{\mathbf{c}}(\mathbf{q}), \tag{6.49}$$

where $\dot{\mathbf{c}}(\mathbf{q}) = \mathbf{D}^T\mathbf{v}$ and $\mathbf{R}$ is a positive definite matrix (usually diagonal). This usually means adding a viscous drag force term, generated by the dissipative potential: $\mathbf{f}_d = -\nabla_{\mathbf{v}}\varphi = -\eta\mathbf{D}\dot{\mathbf{c}}(\mathbf{q})$, where η is a damping coefficient. Very important to note is that these forces act only along the constraint directions and so the damping does not look unnatural.

Using the dissipation potential in (6.49) yields a new regularization formula: $\mathbf{c}(\mathbf{q})+\mathbf{C}^{-1}\mathbf{R}\dot{\mathbf{c}}(\mathbf{q})+\mathbf{C}^{-1}\boldsymbol{\lambda} = 0$, which in turns gives a new KKT matrix and a new Schur complement: $\mathbf{A} = h(h\mathbf{1} + \mathbf{C}^{-1}\mathbf{R})\mathbf{D}^T\mathbf{M}^{-1}\mathbf{D} + \mathbf{C}^{-1}$. We exemplify our result in the case where the stiffness matrix is of the form $\mathbf{C} = \kappa\mathbf{1}$ and for a particular form of Rayleigh damping, i.e. $\mathbf{R} = \eta\mathbf{1} = \rho\mathbf{C}$, where $\rho = \eta/\kappa$. In the end the terms $\mathbf{A}$ and $\mathbf{b}$ in (6.27) get replaced by:

$$\mathbf{A} = h(h+\rho)\mathbf{D}^T\mathbf{M}^{-1}\mathbf{D} + \mathbf{C}^{-1}, \tag{6.50}$$

$$\mathbf{b} = \mathbf{c}(\mathbf{x}) + \rho\mathbf{J}\mathbf{v}. \tag{6.51}$$

Note that damping can also be applied in the case of infinite stiffness $\kappa \to \infty$ (i.e. $\mathbf{C}^{-1} = 0$) given that the ratio ρ remains finite. More recently we came upon a similar result in [TANN13] that also adds support for plasticity.

6.10 Stability

Regularization is equivalent to the following implicit integration scheme at velocity level:

$$(\mathbf{M} + h^2\kappa\mathbf{J}^T\mathbf{J})\mathbf{v}^{l+1} = \mathbf{M}\mathbf{v}^l - h\kappa\mathbf{J}^T\mathbf{c}(\mathbf{x}^l), \tag{6.52}$$

where the last term represents the elastic force and $\tilde{\mathbf{K}} = \kappa\mathbf{J}^T\mathbf{J}$ is an approximate stiffness matrix (the same formulation is used in Section 5 of [AH04a]). But the actual stiffness matrix is [BW98]:

$$\mathbf{K} = -\kappa(\mathbf{J}^T\mathbf{J} + \nabla^2\mathbf{c}(\mathbf{x})\mathbf{c}(\mathbf{x})) = -\tilde{\mathbf{K}} - \kappa\mathbf{H}\mathbf{c}. \tag{6.53}$$

As in [TNGF15] we can denote the second term as the geometric stiffness matrix: $\bar{\mathbf{K}} = \mathbf{H}\lambda$. Given that this part of the stiffness matrix is not implicitly taken into account, it is easy to see now why the regularized velocity time stepping scheme might manifest instabilities above certain frequencies (eigenvalues of $\bar{\mathbf{K}}$). This is mainly due to the linearization error - basically the second order term in the Taylor series expansion of the constraint is missing: $\frac{1}{2}\mathbf{v}^T\mathbf{H}\mathbf{v}$. For the infinite stiffness case we can still have residual error in the solver that can manifest itself as compliance or possibly unstable linearization error. Usually the former is more common for contacts, while the latter can manifest more for bilateral constraints and transverse oscillations modes in threads or cloth.

The solution in [TNGF15] is to add the geometric stiffness term to the mass matrix, but this will make the resulting matrix hard to invert. This is why we chose to continue using a matrix free formulation at the price of running a nonlinear projection solver that deals with the linearization error and also with stability. The case for running more nonlinear iterations for stability is also made in [KTS+14]. We argue that the computational price is only marginally bigger (around 10%) and in return we get better stability.

6.11 Constraint based FEM

The current approach for mixing constraints and stiff elastic forces is to use linear implicit integration and the KKT matrix form of linearized velocity time stepping:

$$\begin{pmatrix} \mathbf{M} - h^2\mathbf{K} & -h\mathbf{J}^T \\ \mathbf{J} & \mathbf{0} \end{pmatrix} \begin{pmatrix} \mathbf{v}^{l+1} \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \mathbf{M}\mathbf{v}^l + h\mathbf{f}_{ext} \\ -\frac{\gamma}{h}\mathbf{c}(\mathbf{x}^l) \end{pmatrix}, \tag{6.54}$$

where $\mathbf{K}$ is the tangential stiffness matrix introduced in Section 4.3. The whole system is then solved using a linear system solver or a mixed LCP solver when unilateral constraints are also present. We take a different approach proposed in [SLM06] based on the regularization concepts described above. For velocity-based DAE solving this can be summarized as:

$$\begin{pmatrix} \mathbf{M} & -h\mathbf{J}^T \\ \mathbf{J} & \mathbf{C}^{-1} \end{pmatrix} \begin{pmatrix} \mathbf{v}^{l+1} \\ \boldsymbol{\lambda} \end{pmatrix} = \begin{pmatrix} \mathbf{M}\mathbf{v}^l + h\mathbf{f}_{ext} \\ -\frac{\gamma}{h}\mathbf{c}(\mathbf{x}^l) \end{pmatrix}, \tag{6.55}$$

that we extended to position based dynamics in Section 6.8.

For the particular case of the geometrically linear *finite element method* (FEM) we use the element energy in order to identify the constraint function: $V_e = W\tilde{\boldsymbol{\varepsilon}}^T\mathbf{C}\tilde{\boldsymbol{\varepsilon}}$, where W is the element volume, $\tilde{\boldsymbol{\varepsilon}}$ is the element constant strain in Voigt notation and $\mathbf{C}$ is a stress-strain relation matrix depending on elastic material properties, i.e. Young's modulus and Poisson ratio (or Lamé coefficients alternatively). For more details see equation (3.16) and the whole Section 3.5. The constraint function is then found by comparing V_e to (6.46):

$$\mathbf{c}(\mathbf{x}) = \sqrt{W(\mathbf{x})}\tilde{\boldsymbol{\varepsilon}}(\mathbf{x}). \tag{6.56}$$

For the strain function we can use any material model we want; just like in [SLM06] we use the nonlinear Green-Lagrange strain, as it preserves the volume under large deformations (in contrast to Cauchy strain). You can find the Jacobian for the constraint in (6.56) further below. We emphasize the fact that even though the the finite elements are linear, we are using a nonlinear St. Vennant-Kirchoff elasticity model.

The advantage of the formulation in (6.55) over (6.54) is that we can invert both $\mathbf{M}$ and $\mathbf{C}$ quite easily, so we can take the Schur complement. In the case of FEM we can use this together with an iterative solver to obtain a matrix-free formulation (in the sense that we don't build the whole system matrix $\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T + h^{-2}\mathbf{C}^{-1}$). If using a Gauss-Seidel or Jacobi approach the whole constraint based FEM method boils down to solving the following 6 by 6 system several times per element:

$$(\mathbf{A}_e + \tfrac{1}{h^2}\mathbf{C}_e^{-1})\boldsymbol{\lambda}_e + \mathbf{b}_e = 0. \tag{6.57}$$

In Algorithm 4 you can find a pseudo-code implementation of the inner loop of a constraint based Gauss-Seidel FEM solver. It can be easily turned into a position based method by updating the positions at every iteration and hence updating the deformation gradient, the strain, the constraint Jacobian and the system matrix. The output of the procedure consists of the 4 forces that will be applied to the tetrahedron vertices. Note that we solve for all the 6 Lagrange multipliers in a block matrix fashion using a direct solver.

Algorithm 4 Pseudo-code for computing the internal forces inside a tetrahedron. Here a block Gauss-Seidel approach is employed.

Input: tetrahedron $(\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$
Compute shape matrix $\mathbf{D}_s = [\mathbf{x}_1 - \mathbf{x}_0 | \mathbf{x}_2 - \mathbf{x}_0 | \mathbf{x}_3 - \mathbf{x}_0]$
Compute deformation gradient $\mathbf{F} = \mathbf{D}_s\mathbf{D}_m^{-1}$
Compute Green strain $\boldsymbol{\varepsilon}$ from the matrix $\frac{1}{2}(\mathbf{F}^T\mathbf{F} - \mathbf{1}_3)$
Compute strain Jacobian $\mathbf{J}$
Compute local system matrix $\mathbf{A} = h^2\mathbf{J}\mathbf{M}\mathbf{J}^T + \mathbf{C}^{-1}$
Solve $\mathbf{A}\boldsymbol{\lambda} + \boldsymbol{\varepsilon} = 0$
Output: internal forces $\mathbf{f} = \mathbf{J}^T\boldsymbol{\lambda} = (\mathbf{f}_0, \mathbf{f}_1, \mathbf{f}_2, \mathbf{f}_3)$

The constraint function in (6.56) depends on $\mathbf{x}$ which is made up of the four tetrahedron vertices: $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$. There are 6 constraints corresponding to the 6 unique values making up the symmetric strain tensor. Let $\boldsymbol{\Lambda}$ be a helper matrix of the same dimension as the Jacobian (6x12). It too can be

split in two 3×12 matrices $\mathbf{\Lambda}_n$ and $\mathbf{\Lambda}_s$. The normal one is:

$$\mathbf{\Lambda}_n = \begin{bmatrix} \mathbf{\Lambda}_n^0 & \mathbf{\Lambda}_n^1 & \mathbf{\Lambda}_n^2 & \mathbf{\Lambda}_n^3 \end{bmatrix}, \tag{6.58}$$

where $\mathbf{\Lambda}_n^i = \text{diag}(\mathbf{y}_i)$, $\mathbf{y}_1$ to $\mathbf{y}_3$ are the rows of a matrix $\mathbf{X}$ and $\mathbf{y}_0 = -\mathbf{y}_1 - \mathbf{y}_2 - \mathbf{y}_3$ [MSJT08]. The matrix $\mathbf{X}$ is the inverse of the initial shape matrix $\mathbf{D}_s$. We are assuming linear finite elements, i.e. tetrahedra of constant strain. For shear we define a similar helper matrix:

$$\mathbf{\Lambda}_s = \tfrac{1}{2} \begin{bmatrix} \mathbf{\Lambda}_s^0 & \mathbf{\Lambda}_s^1 & \mathbf{\Lambda}_s^2 & \mathbf{\Lambda}_s^3 \end{bmatrix}, \tag{6.59}$$

where

$$\mathbf{\Lambda}_s^i = \begin{bmatrix} 0 & y_{iz} & y_{iy} \\ y_{iz} & 0 & y_{ix} \\ y_{iy} & y_{ix} & 0 \end{bmatrix}. \tag{6.60}$$

The normal and shear Jacobians of $\mathbf{c}$ are then:

$$\mathbf{J}_\alpha = \sqrt{W}\mathbf{\Lambda}_\alpha \mathbf{F}^T + \frac{1}{2\sqrt{(W)}} \boldsymbol{\varepsilon}_\alpha \nabla W, \tag{6.61}$$

where $\alpha \in \{n, s\}$, $\mathbf{F} = \mathbf{D}_m \mathbf{X}$ is the deformation gradient, $\boldsymbol{\varepsilon}_n = (\varepsilon_{11}, \varepsilon_{22}, \varepsilon_{33})$ and $\boldsymbol{\varepsilon}_s = (\varepsilon_{23}, \varepsilon_{13}, \varepsilon_{12})$. For Cauchy strain these Jacobians simplify to:

$$\mathbf{J}_\alpha = \sqrt{W}\mathbf{\Lambda}_\alpha + \frac{1}{2\sqrt{(W)}} \boldsymbol{\varepsilon}_\alpha \nabla W, \tag{6.62}$$

The gradient of the volume is then a 12 component row vector consisting of the following partial derivatives:

$$\nabla W = \left(\frac{\partial W}{\partial \mathbf{x}_0} \frac{\partial W}{\partial \mathbf{x}_1} \frac{\partial W}{\partial \mathbf{x}_2} \frac{\partial W}{\partial \mathbf{x}_3} \right), \tag{6.63}$$

where

$$\frac{\partial W}{\partial \mathbf{x}_1} = \frac{1}{6}(\mathbf{x}_2 - \mathbf{x}_0) \times (\mathbf{x}_3 - \mathbf{x}_0), \tag{6.64}$$

$$\frac{\partial W}{\partial \mathbf{x}_2} = \frac{1}{6}(\mathbf{x}_3 - \mathbf{x}_0) \times (\mathbf{x}_1 - \mathbf{x}_0), \tag{6.65}$$

$$\frac{\partial W}{\partial \mathbf{x}_3} = \frac{1}{6}(\mathbf{x}_1 - \mathbf{x}_0) \times (\mathbf{x}_2 - \mathbf{x}_0), \tag{6.66}$$

$$\frac{\partial W}{\partial \mathbf{x}_0} = -\frac{\partial W}{\partial \mathbf{x}_1} - \frac{\partial W}{\partial \mathbf{x}_2} - \frac{\partial W}{\partial \mathbf{x}_3}. \tag{6.67}$$

We have now all the information needed to build the Jacobian $\mathbf{J}^T = (\mathbf{J}_n^T \, \mathbf{J}_s^T)$.

The same formulation can be found in Hammad Mazhar's thesis where it was implemented in a VTS context and also validated [Maz16]. Our main contribution is that we extended the method in [SLM06] to the position projection case by plugging in regularized constraints into the general projection formulation in (6.9). Also by our equivalence result in Section 6.8 we are able to prove that there is no difference between traditional FEM using implicit integration and our proposed method based on constraints and position projection. Again, any VTS variant of this FEM solver is just a semi-implicit linearized version of the nonlinear constraint case.

Damping can be added through the method described in Section 6.9, but other methods can be used too. For example air drag can be formulated directly in the integrator as described in [Jak01]. We used the framework described in this chapter and the FEM solver in particular in [FM15a] to simulate cloth accurately. We think our method is a constraint based contender to implicit integration methods like the one in [VMTF09].

We can apply the block local solve formula (6.57) to the 2D continuum mechanics formulation given in Section 5.4. We choose to apply this block approach only for the stretch components together, as the shear stress component is related only through a diagonal term to strain, and thus decoupled from the normal directions. The resulting 2×2 local linear system for the two stretching constraints is:

$$(h^2\mathbf{A} + \tilde{\mathbf{C}}_{2D}^{-1})\delta\boldsymbol{\lambda} + \hat{\boldsymbol{\varepsilon}}(\mathbf{x}) = 0,$$

where in the case of isotropic materials

$$\tilde{\mathbf{C}}_{2D}^{-1} = \frac{1}{E\sqrt{a}} \begin{pmatrix} 1 & \nu \\ -\nu & 1 \end{pmatrix}.$$

We also implemented volumetric deformable bodies using this theory (for a showcase see Chapters 5 and 8). We argue that our method is physically correct and more accurate than strain limiting techniques [MCKM14] or the continuum PBD method in [BKCW14]. Also, as already stated, we are not far from the Projective Dynamics approach [BML+14] as the problem formulation is the same. We chose to use regularization in order to be able to extract a Schur complement and make our solver matrix-free by applying an iterative method like relaxation or one of the other gradient descent methods described in Section 6.6.

6.12 Unilateral constraints

We used the SQP method throughout this chapter to justify linearized models for a reason, even if it was not always necessary. In the context of bilateral constraints SQP is equivalent to Newton's method, as the problem can be transformed to an unconstrained one. But SQP becomes relevant when adding inequality constraints and at every iteration we can consider projection as a QP with both equality and inequality constraints:

$$\begin{aligned} \text{minimize } & \tfrac{1}{2h^2}\delta\mathbf{x}^T\mathbf{M}\delta\mathbf{x} \\ \text{subject to } & \boldsymbol{\Psi}(\mathbf{x}_{k+1}) = 0, \\ & \boldsymbol{\Phi}(\mathbf{x}_{k+1}) \geq 0, \end{aligned} \tag{6.68}$$

where the constraint function $\mathbf{c}$ was split into a bilateral part $\boldsymbol{\Psi}$ and a unilateral one $\boldsymbol{\Phi}$. The most relevant example of a unilateral constraint is contact. When writing the KKT optimality conditions for (6.68) we obtain the complementarity slackness conditions for the inequality constraints [WN99]:

$$\boldsymbol{\Phi}(\mathbf{x}_{k+1}) \geq 0, \boldsymbol{\lambda}_{\mathcal{U}} \geq 0, \boldsymbol{\lambda}_{\mathcal{U}}^T\boldsymbol{\Phi}(\mathbf{x}_{k+1}) = 0, \tag{6.69}$$

where $\boldsymbol{\lambda}_{\mathcal{U}}$ are the Lagrange multipliers corresponding to unilateral constraints. Two vectors $\mathbf{x}$ and $\mathbf{y}$ are said to be complementary $\mathbf{x} \perp \mathbf{y}$ if they are neither positive nor zero both at the same time for each component. Thus (6.69) can be written alternatively in a shorter form as:

$$0 \leq \boldsymbol{\Phi}(\mathbf{x}_{k+1}) \perp \boldsymbol{\lambda}_{\mathcal{U}} \geq 0. \tag{6.70}$$

This relation is nothing else but the Signorini contact condition [WL06]. This condition states that when the distance between bodies is strictly positive the reaction force is zero, whereas when the distance is zero (the bodies are touching) the force is strictly positive, meaning there cannot be attractive forces, but only repulsive ones. In the latter case the constraints are called active and they correspond to the active set of constraints [WN99]: $\boldsymbol{\Phi}^i(\mathbf{x}) = 0, i \in \mathcal{G}_A$. Note that the complementarity that is included in this condition is already implied in the QP formulation (6.68) where only the inequalities on $\boldsymbol{\Phi}$ are present.

Note that most of the topics covered in this chapter also apply to unilateral constraints, like minimization structure, regularization, damping, stability etc. The only difference is that they now have to be expressed in constrained optimization form or as a complementarity problem. They are more complex than a DAE as not all rows have an equal sign, but also inequality and perpendicularity symbols. Potentially these formulations can be converted to equations and solved with Newton's method [Jea99, BETC14]. However, we chose to use specialized iterative solvers for the inequality problem that are similar to the ones presented in Section 6.6 and that rely on projection operators, e.g. projected Gauss-Seidel (not to be confused with projection on the constraint manifold).

Chapter 7

Nonsmooth dynamics

This chapter is about contact and friction essentially. The name nonsmooth dynamics is given by the fact that both impacts and friction cause discontinuities in the velocity. This does not play well with the classical theory of integration of differential equations. Integrators need to be restarted every time there is an abrupt change in velocity or position and the new values are used as new initial conditions. Event based simulators actually do this by bisecting for the exact time of discontinuity, applying a different algorithm to handle it (e.g. impact law) and then restarting the integrator. But this approach can lead to paradoxes: for example a ball falling on a table with inelastic collision restitution. The time between two successive impacts will become infinitely small (a form of Zeno's paradox) and the number of impacts infinitely high to be handled by a computer.

We are not taking the event based approach here, but a time stepping one: discontinuities are stepped over. Nonsmooth dynamics was developed in the 80s by Moreau, Monteiro-Marques and others precisely to handle the paradoxes of Zeno and Painlevé. Acceleration based methods can handle inequality constraints, contact and friction and they were among the first to exploit the complementarity problem. Switching to using impulses (integrals of force over short times) as the main variables is what lead to the development of velocity time stepping schemes which are implemented in the physics engines of today (see Section 2.2 for a more detailed history).

However, the field is under development and a lot of results are missing. For example the numerical methods for solving frictional contact cannot be proven under all circumstances to converge towards the physical solution of the problem. We also have to keep in mind that a lot of idealization is going on here, but even a simple friction model like Coulomb's can pose serious challenges. And this is the context where we are making our own contribution, namely that position based methods can also treat contact and friction in the framework of nonsmooth dynamics.

7.1 Mathematical prelude

We will introduce now a few notions of convex and nonsmooth analysis that may help you understand the following sections. Unfortunately the field is quite broad to be covered here and also quite complex and still under development. We will give only a few notions that are actually used in this thesis and refer the reader to the many references mentioned above for a more detailed view.

Let us start with the problem formulation by taking the DAE in (6.4)-(6.5) and replacing the bilateral constraint with a unilateral one $\mathbf{\Phi}$ as described in Section 6.12:

$$\mathbf{M}\frac{d\mathbf{v}}{dt} = \mathbf{f}_{ext} + \mathbf{J}^T\boldsymbol{\lambda}, \tag{7.1}$$

$$0 \leq \mathbf{\Psi}(\mathbf{q}) \perp \boldsymbol{\lambda} \geq 0. \tag{7.2}$$

This set of equations is called a *differential complementarity problem* (DCP) [ST96] as it combines a differential equation with a complementarity condition. Note that in this chapter we will use the symbol $\mathbf{q}$ for generalized positions to emphasize the fact that we are now including rotational degrees of freedom too as needed for rigid bodies (see Section 5.1). Also for the velocity derivative we did not use the dot symbol in order to stress the fact that the velocity is no longer considered a smooth function but a function of bounded variation and that $d\mathbf{v}$ is actually a differential measure [Mor88].

The fact that the derivative is not uniquely determined at sharp corners

results in a reformulation of Newton's second law as a set inclusion rather than an inequality. This means that that the acceleration has no longer a clear meaning and the variation of the momentum is caused by *set valued forces*. Also, abrupt changes in the velocity are caused by impulses, which can be seen as integrals of strong forces over short times, i.e. distributions or generalized functions, e.g. the Dirac δ function. This is technically called a *measure differential inclusion* (MDI). In the case of constraints the reaction forces are also potentially not unique (again at sharp corners of the constraint manifold) and so, in order to express them, we need concepts like the indicator function, subdifferential, the subdifferential of the indicator function for the permissible set, normal cone and others. We are not introducing all these concepts here formally and refer the reader to the bibliography. Most of the times we can continue using equality formulations as you will notice throughout this chapter and in many of the references.

The nonlinear complementarity condition in (7.2) can also be reformulated as a *variational inequality* (VI) [CPS92, AB08]:

$$\forall \mathbf{y}, (\mathbf{y} - \mathbf{x})^T \boldsymbol{\Psi}(\mathbf{x}) \geq 0. \tag{7.3}$$

The solution $\mathbf{x}^*$ correspond to $\boldsymbol{\lambda}$ in (7.2). The name variational inequality originates from the elastostatic contact of continua where the variation of a potential function is no longer zero but subject to an inequality given by a unilateral constraint [WL06]. Combining the VI with the constrained equations of motion we obtain a *differential variational inequality* (DVI) [PS08]. This is equivalent to the DCP and it is a popular name given nowadays to the problem of contact and friction as it can also encompass the infinite dimensional case of continuous materials. In this thesis we are only dealing with the finite dimensional case of particles and rigid bodies. In the case of deformable bodies we consider the nodes obtained after finite element discretization (or any other discretization) as particles with lumped mass.

The *normal cone* to a given set S at a point $\mathbf{x} \in S$ is the set:

$$\mathcal{N}_S(\mathbf{x}) = \{\mathbf{p} : \forall \mathbf{y} \in S, (\mathbf{y} - \mathbf{x})^T \mathbf{p} \leq 0\}. \tag{7.4}$$

For an interior point this set is $\{\mathbf{0}\}$. For a boundary point $\mathbf{x} \in \partial S$ the normal cone can be understood as the normal vector if the boundary is smooth or the bundle of all possible normal-like directions when expressed in a sharp corner [Stu09].

The dual of a cone K is the convex cone [BV04]:

$$K^* = \{\mathbf{y} : \forall \mathbf{x} \in K, \mathbf{x}^T\mathbf{y} \geq 0\}. \tag{7.5}$$

The negative normal cone is called the *polar cone* or the tangent cone: $K^\circ = -K^*$ with $K = \mathcal{N}_S$.

7.2 Continuous setting

In order to paint a clear picture of contact dynamics we illustrate in Figure 7.1 a particle contact point with a surface. At this point one can identify a normal to the surface, $\mathbf{n}^i$, and any two tangent vectors, $\mathbf{s}^i$ and $\mathbf{t}^i$, so that together they form an orthonormal frame. When switching to generalized coordinates these normal tangent directions become the vectors $\mathbf{D}_n^i$, $\mathbf{D}_s^i$ and $\mathbf{D}_t^i$ [Ani06, TA11]. Keep in mind that that all pairwise rigid body collisions can be reduced to this scenario. In general, multiple contact points are considered and this why we use i as the contact index.

The most general formulation of constrained dynamics with contact and friction is given in continuous form by a DVI [TA11]:

$$\mathbf{M}\frac{d\mathbf{v}}{dt} = \sum_{i\in\mathcal{G}_A}(\gamma_n^i\mathbf{D}_n^i + \gamma_s^i\mathbf{D}_s^i + \gamma_t^i\mathbf{D}_t^i) + \sum_{i\in\mathcal{G}_B}(\gamma_B^i\nabla\Psi^i) + \mathbf{f}_{tot}^l, \tag{7.6}$$

$$\frac{d\mathbf{q}}{dt} = \mathbf{v}, \tag{7.7}$$

$$\Psi^i(\mathbf{q}) = 0, i \in \mathcal{G}_B, \tag{7.8}$$

$$0 \leq \Phi^i(\mathbf{q}) \perp \gamma_n^i \geq 0, i \in \mathcal{G}_A, \tag{7.9}$$

$$(\gamma_s^i, \gamma_t^i) = \underset{\sqrt{(\gamma_s^i)^2+(\gamma_t^i)^2}\leq\mu^i\gamma_n^i}{\arg\min} (\mathbf{v})^T(\gamma_s^i\mathbf{D}_s^i + \gamma_t^i\mathbf{D}_t^i), \tag{7.10}$$

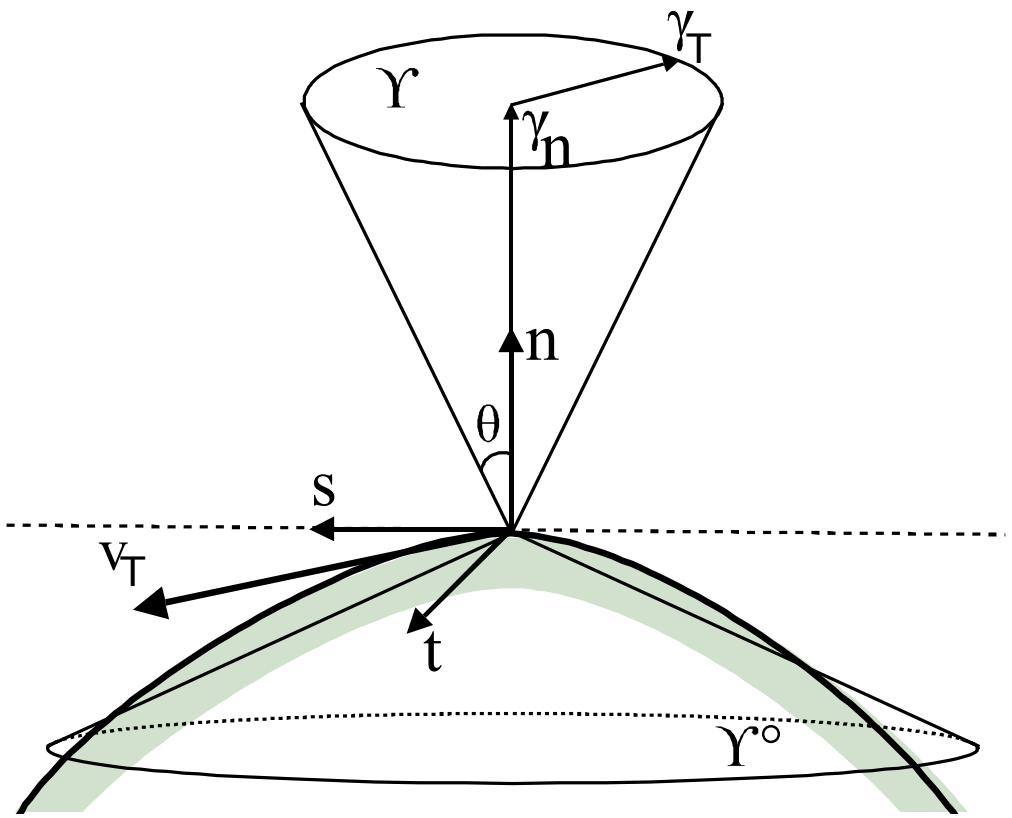


Figure 7.1: Particle contact point with friction cone Υ given by $\theta = \arctan \mu$ and its polar cone Υ° depicted below.

where i is the constraint index, $\mathcal{G}_A$ is the set of active unilateral constraints, $\mathcal{G}_B$ is the set of bilateral constraints, $\Phi^i(\mathbf{q})$ is a unilateral constraint function describing contact (i.e. gap function), $\mathbf{D}_n^i$ is the gradient of the gap function: $\nabla\Phi^i(\mathbf{q}^{l+1})$, γ_n^i is the Lagrange multiplier of the contact condition (7.9), i.e. normal reaction magnitude, $\mathbf{D}_s^i$ and $\mathbf{D}_t^i$ are the generalized tangent directions, γ_s^i and γ_t^i are the corresponding tangent Lagrange multipliers, i.e. friction force components, μ^i is the friction coefficient, $\Psi^i(\mathbf{q})$ is a bilateral constraint function, $\nabla\Psi^i(\mathbf{q})$ is its gradient, γ_B^i is a Lagrange multiplier enforcing bilateral constraints (7.13), and $\mathbf{f}_{tot}$ is the total generalized force acting on the system (external and Coriolis). Note that these equations are an extension to (6.4)-(6.5) as they also accommodate contact (7.9), friction (7.10) and nonsmooth trajectories. We also made a slight change of notation to keep in line with some of our references, namely we use the transpose of the Jacobian

matrices. e.g. $\mathbf{D}_n^i = (\mathbf{J}_n^i)^T$, and denote the Lagrange multipliers by γ and give λ a different meaning (Section 7.3).

We give here directly a discretized form of the DVI:

$$\begin{aligned}\mathbf{M}(\mathbf{v}^{l+1} - \mathbf{v}^l) &= h \sum_{i \in \mathcal{G}_A} (\gamma_n^i \mathbf{D}_n^i + \gamma_s^i \mathbf{D}_s^i + \gamma_t^i \mathbf{D}_t^i) \\ &\quad + h \sum_{i \in \mathcal{G}_B} (\gamma_B^i \nabla \Psi^i) + h\mathbf{f}_{tot}^l, \end{aligned} \tag{7.11}$$

$$\mathbf{q}^{l+1} = \Lambda(\mathbf{q}^l, \mathbf{v}^{l+1}, h), \tag{7.12}$$

$$\Psi^i(\mathbf{q}^{l+1}) = 0, i \in \mathcal{G}_B, \tag{7.13}$$

$$0 \leq \Phi^i(\mathbf{q}^{l+1}) \perp \gamma_n^i \geq 0, i \in \mathcal{G}_A, \tag{7.14}$$

$$(\gamma_s^i, \gamma_t^i) = \underset{\sqrt{(\gamma_s^i)^2 + (\gamma_t^i)^2} \leq \mu^i \gamma_n^i}{\arg\min} (\mathbf{v}^{l+1})^T (\gamma_s^i \mathbf{D}_s^i + \gamma_t^i \mathbf{D}_t^i)., \tag{7.15}$$

where Λ is an implicit Euler integration operator for the positions. The novelty in our approach is that we are using a full implicit Euler integrator instead of semi-implicit/symplectic Euler [ST96, AP97] and we keep the non-penetration condition at position level as a nonlinear unilateral constraint.

Equations (7.11)-(7.12) represent the implicit Euler integration step (see Sections 4.1 and 5.1 for more details). Equation (7.14) represents the Signorini contact complementarity conditions and (7.15) the maximum dissipation principle that synthesizes the Coulomb friction laws [BETC14]. This can also be stated as the condition that the total contact force $\boldsymbol{\gamma}_A^i = (\gamma_n^i, \gamma_s^i, \gamma_t^i)$ should reside inside the friction cone (see Figure 7.1 for an illustration):

$$\Upsilon_A^i = \left\{ \boldsymbol{\gamma}_A^i : \sqrt{((\gamma_s^i)^2 + (\gamma_t^i)^2)} \leq \mu^i \gamma_n^i \right\}. \tag{7.16}$$

Note that the conditions (7.13) and (7.14) are strongly nonlinear which makes the problem hard to tackle. Also, the problem in (7.11)-(7.15) is nonconvex due to the coupling between the friction and the normal force [AH04b, KSJP08]. Usually nonlinearity is eliminated by constraint linearization [AH04a] (see also Section 6.3). In what follows we will present two new ways of handling the nonlinearity and nonconvexity of the problem.

7.3 Polyhedral friction cone

We can reduce the DVI continuous formulation to a series of solvable mixed LCPs by choosing a set of tangent vectors $\{\mathbf{d}_1 \dots \mathbf{d}_p\}$ that describe a regular polygon as the base of a pyramid approximating the friction cone:

$$\begin{aligned} \mathbf{M}\mathbf{v} - \mathbf{M}\mathbf{v}^l - h\mathbf{f}_{tot}^l - h\sum_{i\in\mathcal{G}_B}(\nabla\Psi_k^i\gamma_B^i) - \\ h\sum_{i\in\mathcal{G}_A}(\gamma_n^i\mathbf{D}_{n,k}^i + \sum_{j=1}^{p}\beta_j^i\mathbf{D}_{j,k}^i) = 0, \end{aligned} \tag{7.17}$$

$$i \in \mathcal{G}_A, 0 \leq \Phi^i(\mathbf{q}_k) + h(\mathbf{D}_{n,k}^i)^T\mathbf{v} \perp \gamma_n^i \geq 0, \tag{7.18}$$

$$0 \leq \lambda^i + (\mathbf{D}_{j,k}^i)^T\mathbf{v} \perp \beta_j^i \geq 0, \tag{7.19}$$

$$0 \leq \mu^i\gamma_n^i - \sum_{j=1}^{p}\beta_j^i \perp \lambda^i \geq 0, \tag{7.20}$$

$$i \in \mathcal{G}_B, \Psi^i(\mathbf{q}_k) + h(\nabla\Psi_k^i)^T\mathbf{v} = 0, \tag{7.21}$$

where $\mathbf{D}_j$ is the generalized coordinates equivalent of $\mathbf{d}_j$, β_j are the friction Lagrange multipliers and λ is a Lagrange multiplier approximating the tangential slip velocity. This formulation cannot be expressed as a convex quadratic program (QP) but only as a mixed LCP and the existence of solutions is given by the copositivity of the LCP system matrix [CPS92].

Note that in order to obtain full implicit integration we need to use a fixed point iteration that converges to the solution $\mathbf{v}^{l+1}$ of the DVI nonlinear problem. This is done by successively recomputing the constraint functions and all the constraint directions at point $\mathbf{q}_k$, where k is the iteration number. The solution of each resulting system is $\mathbf{v}_{k+1}$ and the new positions are $\mathbf{q}_{k+1} = \Lambda(\mathbf{q}^l, \mathbf{v}_{k+1}, h)$. The iteration described above is nothing else than the nonlinear scheme presented in Section 3.6 of [ST96]. We present a sketch of the proof of convergence in the rest of the section.

Let $\mathbf{v}_{k+1} = P(\mathbf{v}_k)$ be a mapping that extracts the velocity part of the solution $(\mathbf{v}_{k+1}, \boldsymbol{\gamma}_n, \boldsymbol{\beta}, \boldsymbol{\lambda})$ of the MLCP in (7.17)-(7.21). The existence of the solution for the MLCP is proven by showing that the matrix of the equivalent LCP is copositive [ST96, AH04a].

It can be shown easily that the solution to the nonlinear complementarity problem (NCP) obtained by replacing (7.18) with

$$0 \leq \Phi^i(\mathbf{q}^{l+1}) \perp \gamma_n^i \geq 0,\ i \in \mathcal{G}_A \tag{7.22}$$

is a fixed point of P, i.e. $\mathbf{v}^* = P(\mathbf{v}^*)$. Using Theorem 2.1 from [AH04a] we can show that P is equivalent to a quadratic problem perturbed on both the left and the right hand sides by the parameters $\mathbf{v}_k$ and $\Gamma^i = \mu^i\lambda^i$, i.e. $QP(\mathbf{v}, \mathbf{\Gamma}) \equiv QP(\mathbf{c}(\mathbf{v}), \mathbf{D}(\mathbf{v}), \mathbf{\Gamma})$:

$$\begin{aligned} &\text{minimize } \tfrac{1}{2}\mathbf{v}^T\mathbf{M}\mathbf{v} - \mathbf{v}^T(\mathbf{M}\mathbf{v}^l + h\mathbf{f}_{tot}^l) \\ &\text{subject to } (\mathbf{D}_{n,k}^i)^T\mathbf{v} + \mu^i(\mathbf{D}_{j,k}^i)^T\mathbf{v} \geq -(\tfrac{1}{h}\phi^i(\mathbf{v}_k) + \Gamma^i), \end{aligned} \tag{7.23}$$

where $\phi(\mathbf{v}) = \mathbf{\Phi}(\mathbf{q}^l + h\mathbf{v}) - h\nabla\mathbf{\Phi}(\mathbf{q}^l + h\mathbf{v})^T\mathbf{v}$. Using Lemma 2.2 from [AH04a] we can choose a solution $\lambda^* = -\min_j\{\mathbf{d}_j^T\mathbf{v}^*\} = \Lambda(\mathbf{v}^*)$, where $\mathbf{v}^*$ is a solution of (7.23) or a fixed point of P. As the mapping P is equivalent to QP we can now write the fixed point formulation:

$$\mathbf{v}^* = P(\mathbf{v}^*) = QP(\mathbf{v}^*, \mu\Lambda(\mathbf{v}^*)). \tag{7.24}$$

We can now prove that P is a contraction or that

$$\|P(\mathbf{v}_1) - P(\mathbf{v}_2)\| \leq c\|\mathbf{v}_1 - \mathbf{v}_2\|, \tag{7.25}$$

for any $\mathbf{v}_1$ and $\mathbf{v}_2$ and $c < 1$. The QP in (7.23) is non-convex in terms of $\mathbf{v}$ and λ, but convex in terms of $\mathbf{v}$ only as the matrix $\mathbf{M}$ is positive definite. Using Theorem 1 from [AP97] we get Lipschitz continuity for the mapping QP and by adapting Theorem 3i from the same source we get:

$$\|QP(\mathbf{v}_1, \Gamma_1) - QP(\mathbf{v}_2, \Gamma_2)\| \leq L(K_\mu, K_\Gamma, K_\mathbf{v})(\|\Gamma_1 - \Gamma_2\| + \|\mathbf{v}_1 - \mathbf{v}_2\|), \tag{7.26}$$

for any $\Gamma_j \leq K_\Gamma$ and $\mathbf{v}_j \leq K_\mathbf{v}$, where $j = 1, 2$ and $\mu \leq K_\mu$. These assumptions are needed as they are tightly connected to the feasibility of the QP in (7.23): this happens only under Mangasarian-Fromovitz constraint

qualification (MFCQ) which is equivalent to a pointed friction cone condition (Lemma 2.5, [AH04a]). We now use the fact that $\Gamma = \mu\Lambda(\mathbf{v})$ and the mapping Λ is globally Lipschitz with a parameter K_D (Lemma 6, [AP97]):

$$\|\Lambda(\mathbf{v}_1) - \Lambda(\mathbf{v}_2)\| \leq K_D\|\mathbf{v}_1 - \mathbf{v}_2\|, \tag{7.27}$$

for any $\mathbf{v}_1$ and $\mathbf{v}_2$, and so from equations (7.24), (7.26) and (7.27) we obtain:

$$\|P(\mathbf{v}_1) - P(\mathbf{v}_2)\| \leq \mu L(K_\mu, K_\Gamma, K_\mathbf{v})(K_D + 1)\|\mathbf{v}_1 - \mathbf{v}_2\|, \tag{7.28}$$

for any $\mathbf{v}_1$ and $\mathbf{v}_2$ in a vicinity of $\mathbf{v}^*$, where $\mu = \max(\mu^i)$ and L, K_μ, K_Γ, $K_\mathbf{v}$ and K_D are Lipschitz continuity parameters. Thus, if we choose μ sufficiently small:

$$\mu < \mu^\circ = \frac{1}{L(K_\mu, K_\Gamma, K_\mathbf{v})(K_D + 1)}, \tag{7.29}$$

we can prove that P is a contraction and so the fixed point iteration in (7.17)-(7.21) converges to a solution of the NCP. Moreover, it can be shown that it is also dissipative by reusing Theorem 3.2 from [AH04a] in a recurrent fashion, i.e. the kinetic energy after any number of iterations is always less than the initial kinetic energy plus the work of external forces.

7.4 Smooth friction cone

In [Ani06] the DVI is linearized and convexified so that it can be expressed in the end as a quadratic minimization problem with conic constraints. We take this formulation and extend it to the fully implicit and nonlinear case in (7.11)-(7.15):

$$\begin{aligned} \text{minimize } & W(\mathbf{v}) = \tfrac{1}{2}\mathbf{v}^T\mathbf{M}\mathbf{v} - \hat{\mathbf{f}}^T\mathbf{v} \\ \text{subject to } & \Phi^i(\mathbf{q}^{l+1}) - h\mu^i\|\mathbf{v}_T^i\| \geq 0, i \in \mathcal{G}_A, \\ & \Psi^i(\mathbf{q}^{l+1}) = 0, i \in \mathcal{G}_B, \end{aligned} \tag{7.30}$$

where $\hat{\mathbf{f}} = \mathbf{M}\mathbf{v}^l + h\mathbf{f}_{tot}^l$ and $\|\mathbf{v}_T^i\| = \sqrt{((\mathbf{D}_s^i)^T\mathbf{v})^2 + ((\mathbf{D}_t^i)^T\mathbf{v})^2}$ is the magnitude of the tangential relative velocity at the contact point. It is easy to show that

the solution to (7.30) is a fixed point of the smooth cone optimization problem in [Ani06] (equation 5.5). Our approach for solving this problem is to derive a new fixed point iteration that is equivalent to a CCP at every kth iteration:

$$\Upsilon_k^\circ \ni -(h\mathbf{D}_k^T\mathbf{v} + \mathbf{b}_k) \perp \boldsymbol{\gamma} \in \Upsilon_k, \tag{7.31}$$

where $\boldsymbol{\gamma} = (\boldsymbol{\gamma}_A, \boldsymbol{\gamma}_B)$ is the Lagrange multipliers vector and $\mathbf{b}_k = (\mathbf{b}_{k,A}, \mathbf{b}_{k,B})$ - the first component corresponding to contacts $\mathbf{b}_{k,A} = (\boldsymbol{\Phi}(\mathbf{q}_k) - h\mathbf{D}_{n,k}^T\mathbf{v}_k\,, 0\,, 0)$ and the second to bilateral constraints $\mathbf{b}_{k,B} = \boldsymbol{\Psi}(\mathbf{q}_k) - h\nabla\boldsymbol{\Psi}_k^T\mathbf{v}_k$. Υ is the direct sum of all friction and bilateral cones, Υ° is the corresponding polar cone and $\mathbf{D}$ is the concatenation of all constraint directions, i.e. $\mathbf{D}_A^i = [\mathbf{D}_n^i|\mathbf{D}_s^i|\mathbf{D}_t^i]$ and $\mathbf{D}_B^i \equiv \nabla\Psi^i$. You can consult [TA11] for more details on notation and how (7.31) can be derived from a linearization of (7.30) around $(\mathbf{q}_k, \mathbf{v}_k)$. The CCP in (7.31) can be solved using another fixed point iteration based on matrix splitting (i.e. relaxation) with constraint projection, shown to converge in [TA11]. After k iterations the solution is $\mathbf{v}_{k+1}$ which can be substituted for $\mathbf{v}^{l+1}$.

We would like to note that the scheme in (7.31) is similar in approach to the nonlinear mixed complementarity scheme presented in Section 3.6 of [ST96]. Also, running only one relaxation iteration inside the Newton like scheme in (7.31) is analogous to the nonlinear projected Gauss-Seidel or Jacobi scheme used in PBD [Jak01, MHHR07]. We will show in the next section that the nonlinear iteration (7.31) is actually the basis for position projection methods (including PBD) and that this is a novel result that allows the accurate inclusion of friction inside position based solvers.

One drawback of the velocity based convexified smooth cone approach is that it may produce normal impulse artifacts [Ani06] but these manifest only for high slip speeds and big friction coefficients [MHNT15]. Given that our approach is a fixed point iteration very similar to the one in [ACLM11] (i.e. the velocity is updated at every iteration) it should also converge to the solution of the original nonconvex problem. However, in practice, when running fewer iterations one may choose to use a different friction model in order to avoid potential artifacts. For example one can use the mixed LCP

(MLCP) polyhedral friction cone model (see supplemental material for a convergence proof) or another model like box LCP (BLCP) friction [BETC14]. Note that all these alternative friction models manifest anisotropy. Another choice was to use a simple cylinder projection operator that does not affect the normal impulse. This simply clamps the friction force at each iteration, but unfortunately we do not a have convergence proof (although in simple stacking experiments we saw no difference to cone projection). For further discussion see Section 7.8.

Extending the proof in the previous section to the smooth cone case boils down to showing that the solution of (7.30) is a fixed point of the mapping in (7.31), i.e. $\mathbf{v}^* = CCP(\mathbf{v}^*)$, and that $CCP(\mathbf{v})$ is a contraction. The former is trivial, while the latter can be shown by noting that $CCP(\mathbf{v})$ is a perturbed CCP that results from the QP formulation from [Ani06] in the limit case $p \to \infty$. Thus we can obtain a Lipschitz continuity (sensitivity) result for the CCP mapping similar to the one presented above. Then if we choose a sufficiently small maximum friction coefficient like the one in (7.29) we can show that the fixed point iteration described in (7.31) converges. Still, this is not a complete result, as we do not have a theorem ensuring stability of the conic constraints problem $CCP(\nu)$ under perturbations ν like we had in the polyhedral case. Even so, we conjecture that our result is true for the smooth cone case and it can be proven in the future with a more thorough mathematical investigation.

7.5 Position projection

If we define $\tilde{\mathbf{v}} = \mathbf{v}^l + h\mathbf{M}^{-1}\mathbf{f}^l_{tot} = \mathbf{M}^{-1}\hat{\mathbf{f}}$ as the unconstrained velocity then it is easy to show that the problem in (7.30) remains unchanged if we set $W(\mathbf{v}) = \frac{1}{2}\delta\mathbf{v}^T\mathbf{M}\delta\mathbf{v}$, where $\delta\mathbf{v} = \mathbf{v} - \tilde{\mathbf{v}}$. By looking at (7.12) we can see that we can approximate $\delta\mathbf{v}$ by a linear relationship to the displacements $\delta\mathbf{q} = \mathbf{q}^{l+1} - \tilde{\mathbf{q}}$, where $\tilde{\mathbf{q}} = \Lambda(\mathbf{q}^l, \tilde{\mathbf{v}}, h)$ are the unconstrained positions. This relation is often exact in implementations and we can write $\delta\mathbf{v} = h^{-1}\mathbf{L}\delta\mathbf{q}$, where $\mathbf{L}$ is a linear mapping that can depend on $\mathbf{q}$. Thus it can be shown that the

quadratic objective in (7.30) can be reformulated in terms of displacements:

$$\begin{aligned} &\text{minimize } W(\delta\mathbf{q}) = \tfrac{1}{2h^2}\delta\mathbf{q}^T\bar{\mathbf{M}}\delta\mathbf{q}, \\ &\text{subject to } -\mathbf{u} \in \Upsilon^\circ, \end{aligned} \tag{7.32}$$

where $\bar{\mathbf{M}} = \mathbf{L}^T\mathbf{M}\mathbf{L}$ and $\mathbf{u}(\delta\mathbf{q}) = (\mathbf{u}_A, \mathbf{u}_B)$ with $\mathbf{u}_A = h\mathbf{D}_k^T\mathbf{v} + \mathbf{b}_k = (\Phi(\mathbf{q}^{l+1}), h\mathbf{D}_s^T\mathbf{v}^{l+1}, h\mathbf{D}_t^T\mathbf{v}^{l+1})$ and $\mathbf{u}_B = \Psi(\mathbf{q}^{l+1})$. The constraints are the same as for (7.30) but they are expressed in a more compact form as in [TA11]: the velocity-like term $\mathbf{u}$ should reside inside the dual of the friction cone $\Upsilon^* = -\Upsilon^\circ$.

In this form we can easily recognize the projection method for solving differential equations on manifolds (see equation (6.9)). The fixed point iteration in (7.31) extends the projection method to unilateral constraints and friction, given that we use the initial guess for velocity $\mathbf{v}_0 = \tilde{\mathbf{v}}$. If we use $\mathbf{v}_0 = 0$ instead and only one fixed point iteration we obtain the velocity time stepping method with linearized constraints that is prevalent in rigid body dynamics simulations [AH04a].

For the equality constrained case, the fast projection method presented in [Gol10] can be used to minimize (7.32). PBD accomplishes the same task using nonlinear projected relaxation (e.g. Gauss-Seidel, Jacobi, SOR). What fast projection essentially does is it successively projects the positions on the constraint manifold. This is done by iteratively solving the weighted least squares problem given by (7.32) in terms of the dual variables $\boldsymbol{\gamma}_{k+1}$. Or rather constraint force increments $\delta\boldsymbol{\gamma}_{k+1}$ are computed in a sequential quadratic programming (SQP) approach, which are then used to update the state of the system. This is the frictionless case presented in Section 6.12.

The dual form problem is obtained following the approach in [MHNT15] and [TA11] (see also Section 6.4):

$$\begin{aligned} &\text{minimize } \tfrac{1}{2}\delta\boldsymbol{\gamma}^T\mathbf{A}_k\delta\boldsymbol{\gamma} + \delta\boldsymbol{\gamma}^T\mathbf{r}_k \\ &\text{subject to } \boldsymbol{\gamma}_k + \delta\boldsymbol{\gamma} \in \Upsilon_k, \end{aligned} \tag{7.33}$$

where $\mathbf{A}_k = h^2\mathbf{D}_k^T\bar{\mathbf{M}}^{-1}\mathbf{D}_k$ and $\mathbf{r}_k = h\mathbf{D}_k^T\mathbf{v}_\mathbf{k} + \mathbf{b}_k$. By taking the limit $k \to \infty$

we can formulate this SQP-like fixed point iteration by a single nonlinear minimization problem with conic constraints:

$$\begin{aligned} &\text{minimize } \tfrac{1}{2}\boldsymbol{\gamma}^T\mathbf{A}\boldsymbol{\gamma} + \boldsymbol{\gamma}^T\mathbf{r} \\ &\text{subject to } \boldsymbol{\gamma} \in \Upsilon, \end{aligned} \tag{7.34}$$

where $\mathbf{A} = h^2\mathbf{D}^T\bar{\mathbf{M}}^{-1}\mathbf{D}$ with $\mathbf{D}$ evaluated in $\mathbf{q}^{l+1}$ and $\mathbf{r} = \mathbf{u}$. This is none more than the dual of (7.32). In conclusion, equations (7.32) and (7.34) are the most general formulations of position projection methods (e.g. fast projection, PBD) and they are derived from the nonlinear velocity based formulation in (7.30). We believe that this is a new result and it makes a powerful connection between velocity and position level methods.

7.6 Projected iterative solvers

All the solvers presented in the previous chapter (Section 6.6) can also be applied to VTS methods and nonlinear position projection too. This is done using projection operators on the permissible set. This was done in the past for relaxation solvers for both VTS (either in a LCP or CCP formulation) [CPS92, TA11, Lac03] and PBD. In essence this was the motivation for all our nonlinear schemes described above: to develop a rigorous mathematical model for projection on the friction cone. This is because in all our investigations of the literature we found no clear way of handling friction in the context of position based dynamics and no proof to why such a nonsmooth dynamics approach would work. We think this is an important contribution that was lacking - up to this point we found no previous work that stated the same results in a direct fashion. Also, this is also the strongest proof besides the regularization equivalence approach in the previous chapter that position projection is indeed a physically correct method.

We followed the example of projected relaxation methods and devised our own general projected gradient descent template for solving unilateral

constraints similar to (6.33) [FM14a]:

$$\boldsymbol{\lambda}_{k+1} = \text{proj}(\boldsymbol{\lambda}_k + \alpha \mathbf{d}_k + \beta(\boldsymbol{\lambda}_k - \boldsymbol{\lambda}_{k-1})), \tag{7.35}$$

where the projection operator can be as simple as a clamp for nonnegative or box constraints [SHNE10b, Lac03] or more complex like a cone projection [TA11]. Our mathematical approach was not so mathematically rigorous but we found it to work in practice. For example we chose not to do gradient projection as is done in other works [RA05, HATN12, WN99] but only project the solutions at every iteration on the admissible set. The motivation behind this was that in the end we developed an improved or accelerated form of Jacobi that is not that different from the classical method, so neither should be the projection step. However, more mathematical investigation should be done here to see if additional tangent space projection steps are beneficial. The last term in (7.35) is called a momentum term and it is used mainly in APGD [MHNT15] and our improved Jacobi scheme.

7.7 Rigid bodies

In this section we present another of our contributions: a rigid body simulation method with contact and friction using position projection. Previous position based contact approaches are not very rigorous [Jak01, MHHR07]. Some authors also preferre to use bilateral constraints instead [Gol10, Bri14]. The only paper we found on rigid body simulation using position based dynamics [DCB14] does not offer a proof to why the method works and there is no backing material in the mechanical engineering literature either.

Modeling contact and friction can be done by adding a gap scalar function along the contact normal direction $\mathbf{n}_{ij}$:

$$\Phi(\mathbf{q}) = \mathbf{n}_{ij} \cdot (\mathbf{x}_i + \mathbf{R}_i \mathbf{p}_i - \mathbf{x}_j - \mathbf{R}_j \mathbf{p}_j), \tag{7.36}$$

where $\mathbf{p}_i$ and $\mathbf{p}_j$ are the pair of closest points expressed in their respective local frames. The derivation of the Jacobian, i.e. the normal generalized

constraint direction $\mathbf{D}_n$, can be found in many references [Cat05, Smi05], while $\mathbf{D}_s$ and $\mathbf{D}_t$ can be built from it. Similarly a bilateral constraint representing a spherical joint can be represented by a 3 valued vector function: $\Psi(\mathbf{q}) = \mathbf{x}_i + \mathbf{R}_i\mathbf{p}_i - \mathbf{x}_j - \mathbf{R}_j\mathbf{p}_j = 0$. A hinge constraint has only one value, this number representing the number of unconstrained rotational degrees of freedom.

Contact only has been tackled in the past either by instantaneously considering it as a bilateral constraint or through a crude complementarity approach. Friction on the other hand has had no solid mathematical framework to rely on and we believe that our nonlinear fixed point iteration is the first (using either an LCP or CCP discretization). You can find our pseudo-code for frictional contact between rigid bodies in Algorithm 5. Note that we identify the two bodies by the indices 1 and 2 and a contact pair is fully determined by a world normal $\mathbf{n}$ and the closest points between the two bodies $\mathbf{a}_1$ and $\mathbf{a}_2$ - each expressed in their respective frame.

Algorithm 5 Pseudo-code for computing the normal and friction forces between 2 rigid bodies in contact. Can be used with either a Jacobi or a Gauss-Seidel approach ($\omega \geq 1, \beta = 0$).

Input: contact pair $(\mathbf{n}, \mathbf{a}_1, \mathbf{a}_2)$, β, old force $\boldsymbol{\gamma}$, and increment $\delta\boldsymbol{\gamma}$
$\mathbf{p}_1 = \mathbf{R}_1\mathbf{a}_1, \mathbf{p}_2 = \mathbf{R}_2\mathbf{a}_2$
Compute normal residual $r_n = \mathbf{n} \cdot (\mathbf{x}_1 + \mathbf{p}_1 - \mathbf{x}_2 - \mathbf{p}_2)$ (gap)
Compute normal diagonal term d_n of matrix $\mathbf{A}$
$\gamma_n = \text{clamp}(\gamma_n - \frac{\omega}{h^2 d_n} r_n - \beta\delta\gamma_n, 0, \infty)$, $\gamma_T = 0$
Compute relative velocity $\mathbf{v}_{12} = (\mathbf{v}_1 + \boldsymbol{\omega}_1 \times \mathbf{p}_1) - (\mathbf{v}_2 + \boldsymbol{\omega}_2 \times \mathbf{p}_2)$
Compute tangential relative velocity $\mathbf{v}_T = \mathbf{v}_{12} - (\mathbf{n} \cdot \mathbf{v}_{12})\mathbf{n}$
if $\mathbf{v}_T \neq 0$ **then**
    Compute tangential residual $r_T = \|\mathbf{v}_T\|$ (slip speed)
    Compute tangential direction $\boldsymbol{\tau} = \mathbf{v}_T / v_T$
    Compute tangential diagonal term d_T
    $(\gamma_n, \gamma_T) = \text{project}(\gamma_n, \gamma_T - \frac{\omega}{h^2 d_T} r_T - \beta\delta\gamma_T)$
end if
Output: contact force $\boldsymbol{\gamma} = (\gamma_n, \gamma_T)$, i.e. $\mathbf{f} = \gamma_n\mathbf{n} + \gamma_T\boldsymbol{\tau}$

7.8 Friction models

The pseudocode in Algorithm 5 is built in such a way that we only need one tangential direction for computing the friction impulse, i.e. $\boldsymbol{\tau} = \mathbf{v}_T / v_T$ (although this may prove tricky when v_T is close to zero). We found that when using the cone projection operator in [TA11] we do not necessarily need to use two tangent orthogonal directions and projecting directly the vector along $\boldsymbol{\tau}$ is a good approximation. This is also a good optimization in the sense that we only do the relaxation solve for one normal and one tangential component per contact. The other existing methods need to do at least 3 local solves per contact. Another optimization we did was to use a cylinder projection (i.e. clamp the magnitude of friction below the current normal threshold $\mu\gamma_n$), even though we do not have a proof of convergence for this scheme. The optimizations worked well and yielded plausible simulations. Our motivation behind these optimizations was to simplify the model and thus make it easy to implement for real-time applications where friction is already heavily approximated.

For an overview of friction models see [Pre08] or the discussion in [Lac07]. It is important to note that some of these approximations are using an estimate of the normal impulse component - the case of box or cylinder friction. This estimate can be a constant [Cat05], coming from a previous frame or from a previous solver iteration [PNE10]. The case that running the normal contact solver and friction solver one after the other is actually a staggered approach is made in [BETC14]. The most popular friction model in computer graphics and games is the box LCP or the square pyramid one [Erl07, TBV12]. It was also used in the engineering literature [PT96]. Note though that this is not the same pyramid approximation as in the Stewart-Trinkle model (that would correspond to 4 spanning directions/faces). The square pyramid has its edges parallel to the tangent axes, whereas the latter polyhedral friction pyramid is rotated by 45 degrees (and the friction limits vary depending whether the point is in a vertex or on an edge). However, we have not implemented the Stewart-Trinkle model, as the square pyramid model is better suited for interactive simulations.

Finally, we want to emphasize the fact that our nonlinear fixed point iterative solver, and hence position based dynamics, can handle friction with any of the above friction models, depending on the implementation choices. This is why we provided proofs for the two most general cases we could think of and the other models are just approximations. Still, it would be nice to have clearly written proofs for the square pyramid and cylinder projection that we used in practice. We leave this for future work.

Chapter 8

Unified simulation framework

8.1 Nonlinear constrained dynamics

Our unified simulator relies on the theoretical aspects presented in the previous chapters. It is in essence a position based dynamics solver with several extensions. The most important result is that the problem can be formulated as a constrained minimization. This permits us to use a range of algorithms suited for nonlinear optimization.

The most compressed form of our solver can be found in equation (7.33) with the contact and friction handling from Chapter 7 and bilateral constraints like the ones described in Chapter 6. Note that we can also use regularization and damping to obtain springier soft constraints. This can also be applied to contacts. Also very important, such constraints allow us to simulate deformable bodies using FEM. Friction is handled correctly too: for this we need to update the velocities at the same time as correcting the positions. You can find the outline of such a solver in Algorithm 6 using a Jacobi approach. Note that in the inner loop one can plug any type of constraint solves like the ones in Algorithm 5 and 4. Some two way coupling results between rigid and flexible bodies can be seen in Figures 8.1 and 8.2.

Algorithm 6 Nonlinear projected gradient descent constraint solver using a Jacobi approach.

Unconstrained step to $\tilde{\mathbf{q}}, \tilde{\mathbf{v}}$
$\mathbf{q}_0 = \tilde{\mathbf{q}}, \mathbf{v}_0 = \tilde{\mathbf{v}}$
for k = 0:$k_{max} - 1$ **do**
    Compute $\mathbf{c}(\mathbf{q}_k)$ and $\mathbf{D}_k$
    Compute residual $\mathbf{r}_k$
    Update Lagrange multipliers (see Section 7.6)
    Apply generalized force $\mathbf{f}_c = \mathbf{D}_k \delta\boldsymbol{\gamma}_{k+1}$ using both position and velocity integrators
end for

8.2 Implementation and results

All of the algorithms were implemented in C++ in a unified manner such that all constraints were solved at the same time and in the same solver. For this we used a single common list of bodies that could have each a maximum of 6 degrees of freedom. This list was split into groups, each group having a different meaning (e.g. cloth, rigid body system or FEM soft body) and different types of constraint lists. Some constraint types were specific to only one group (e.g. link constraints for cloth), others were common among several groups (e.g. contact constraints) and the rest were specially designed for coupling between groups (e.g. rigid body vs. triangle).

In terms of constraint solving we used mainly two approaches: Gauss-Seidel and accelerated Jacobi. We further optimized the latter using OpenMP parallel for loop directives. You can find the speed-up factor in comparison to Gauss-Seidel in Tables 8.1 and 8.2. Measurements were done on a dual core laptop CPU (i5-3317U) and a quad core desktop CPU (i7-3770).

For cloth simulation our test model was mostly a rectangular grid of point particles in order to avoid interlocking of triangle elements. We connected the particles using stretch, shear and bend resisting links (see Section 5.4). All the links were modeled as rigid constraints (as described in Chapter 6) with various stiffness values like in [MHHR07]. Stretch constraints had a stiffness factor ω (see equation (6.48)) of 1 or more if using Successive Over

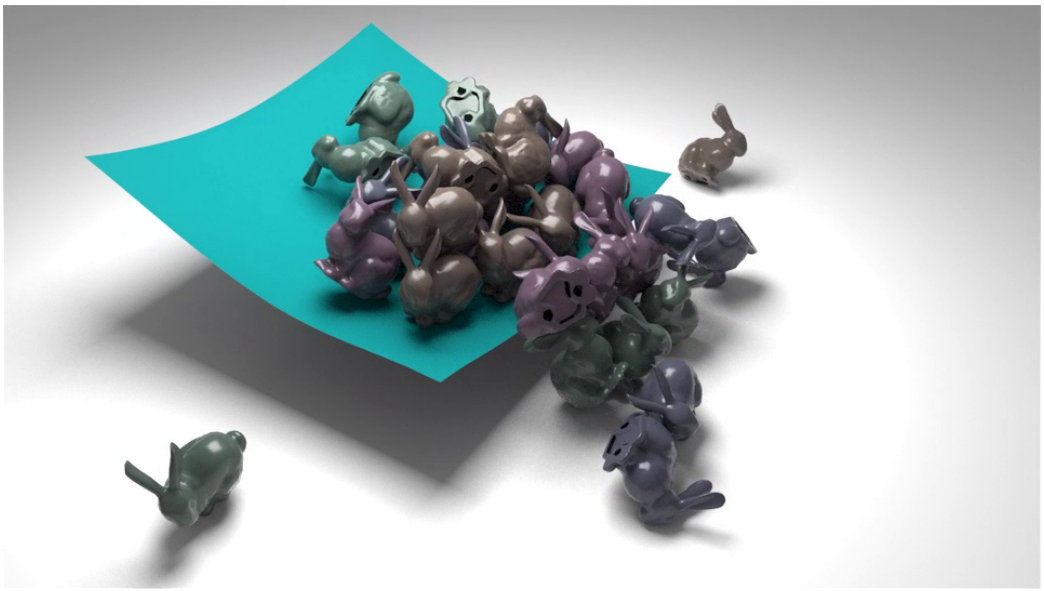


Figure 8.1: Rigid bunnies falling on a piece of cloth with two way coupling.

	Gauss-Seidel	Accelerated Jacobi	Speedup
2000 boxes	140 ms	90 ms	1.55x
50×50 cloth	6.3 ms	2.7 ms	2.33x
100×100 cloth	27.5 ms	19 ms	1.45x

Table 8.1: CPU time (for one simulation frame) comparison between Gauss-Seidel and accelerated Jacobi nonlinear constrained dynamics solvers (dual core).

Relaxation (SOR), whereas the other two types were closer to zero. Shearing and bending links can also be treated as low stiffness springs, and thus can be integrated explicitly. This is equivalent to doing only one projection step at the beginning. If one needs more shearing and bending stiffness then more projection steps need to be interleaved with the stretching ones.

For unconstrained integration we used Verlet, Symplectic Euler or the special Newmark candidate step in Section 6.5. In time we gave up on Verlet as it did not permit us to work directly with velocities.

This cloth model based on hard links is very popular for real-time simulation in games as it is very simple and can be easily optimized. Although we did implement angular bending constraints, one can use only links and

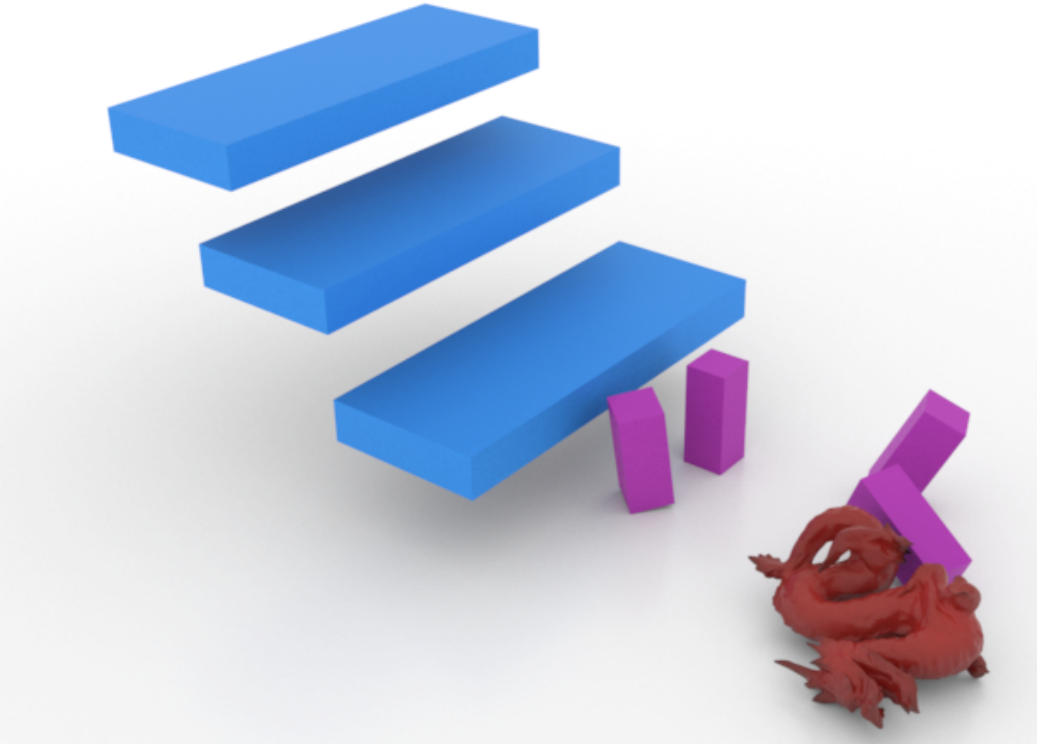


Figure 8.2: Flexible dragon falling on stairs and hitting rigid boxes.

contacts in order to keep the solver inner loop simple. This permitted us to keep the code unified on the GPU despite the fact that bending links do not always look well or are hard to build for triangular meshes. Usually real-time models are not very large (few hundred particles) and are solved using Gauss-Seidel type iterations on the CPU. For larger cloth pieces one needs to exploit the parallel nature of the hardware and this is not trivial for GS. Jacobi and MINRES are better suited for running on multi-core or GPU

	Gauss-Seidel	Accelerated Jacobi	Speedup
2000 boxes	82.5 ms	40 ms	2.06x
100×100 cloth	15.4 ms	3.5 ms	4.4x
150×150 cloth	36.3 ms	15 ms	2.42x

Table 8.2: CPU time (for one simulation frame) comparison between Gauss-Seidel and accelerated Jacobi nonlinear constrained dynamics solvers (quad core).

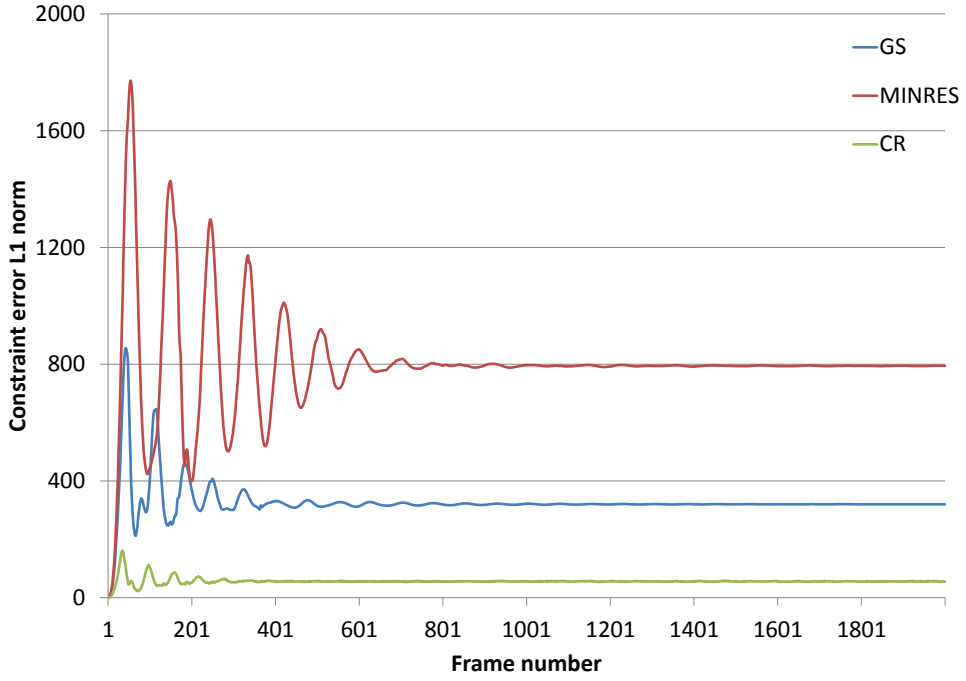


Figure 8.3: Evolution of cloth from initial state to steady state painted as L1 norm of the constraint error for 30 iterations per frame using GS (blue), MINRES (red) and CR (green).

but they need many more iterations. This is solved by CR as it has better convergence rate than GS or by accelerated Jacobi which is comparable to GS.

Our experiments have shown that the CR method converges better than GS for the same number of iterations, thus making up for the extra computational cost. We illustrated stability and convergence information in Figures 8.3 and 6.3 and performance data in Figure 8.4. Measurements were done on a 50×50 piece of cloth simulated with a time step of 8 ms on an Intel Core i7 3770 (single threaded).

As you can see in the plot (Figure 6.3) already from 4 iterations CR has better convergence than GS and it keeps getting better until at around 9 iterations when it's already more CPU efficient to run CR instead of GS. And the gap between the two increases linearly with the number of iterations,

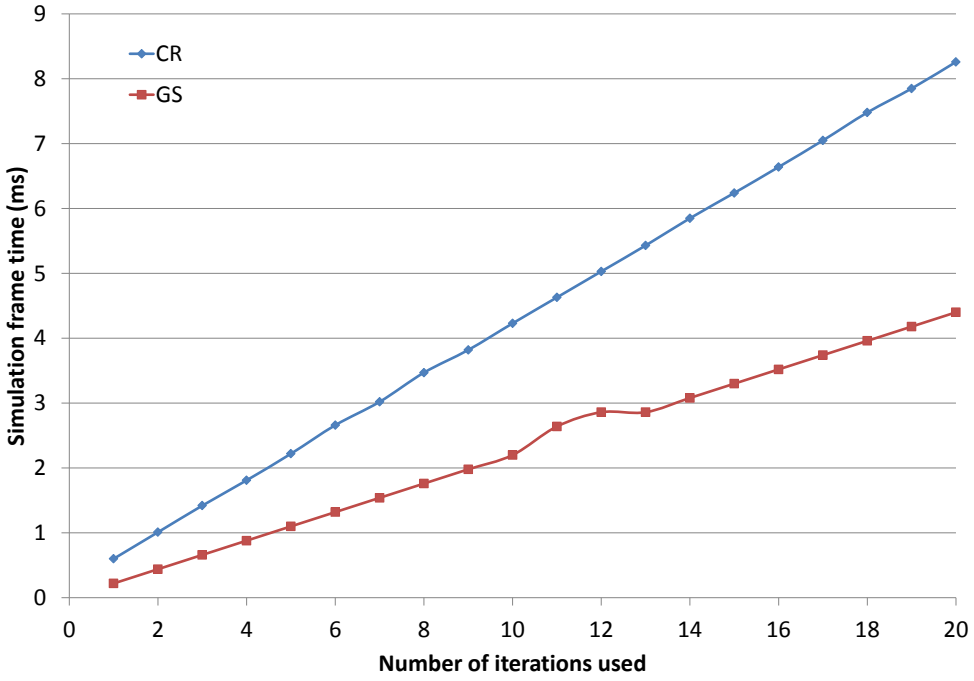


Figure 8.4: Plot of the simulation cost per frame in milliseconds (vertical axis) relative to the number of iterations (horizontal axis) for GS (red) and CR (blue).

thus making CR especially suitable for very large problems targeting a large degree of inextensibility. For example at 35 iterations GS takes roughly 1.5 more time (7.7 ms) than 12 iterations of CR (5 ms) that actually provide a smaller error.

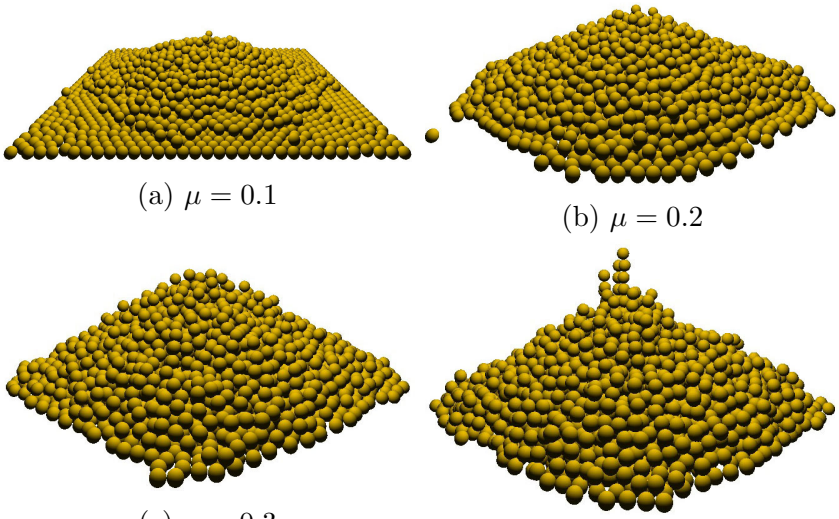
We implemented simplified parallel versions of both MINRES and CR using C++ and OpenCL based on a Jacobi solver template with little modifications. After running this parallel implementation on a multi-core CPU we obtained a speedup of up to 6 times compared to single threaded.

These results were also presented in [FM14b] and later extended in [FM14a] where we introduced the improved form of Jacobi and a first method of realizing coupling and contact damping between particles and cloth: we called it Sequential Positions (SP). In the latter paper we also focused more on granular matter simulation as you can see in Figure 5.9. We also took better care

of friction (Figure 8.5), although we did not have a clear theoretical model and convergence result at that time. We attempted to reproduce the behavior described in Section 5.8 with our methods and you can see the resulting sand piles in Figure 8.6. Notice how the angle of repose increases with the value of the friction coefficient.

(a) (b)

Figure 8.5: Cloth falling freely over a sphere with friction coefficient $\mu = 0.5$: (a) using accurate friction inside the iterative solver the cloth remains stable on the sphere and (b) using the traditional PBD friction handling (velocity post-processing) method the cloth falls very quickly off the sphere.



(a) $\mu = 0.1$ (b) $\mu = 0.2$

(c) $\mu = 0.3$ (d) $\mu = 0.4$

Figure 8.6: Sand piles formed by dropping 3000 particles using VTS with different friction coefficients (15 iterations, Baumgarte stabilization $\gamma = 0.5$).

Initially we tested the accuracy of our improved Jacobi method on bilat-

eral constraints only. Our test scenario consisted of a 100×100 piece of cloth falling from a horizontal position and hanging from two corners. The simulation used a PBD method with a time step of 16 ms, one sub-step and 15 iterations. You can see the evolution in time of the system for three different solvers in Figure 8.7. Clearly our method performs better.

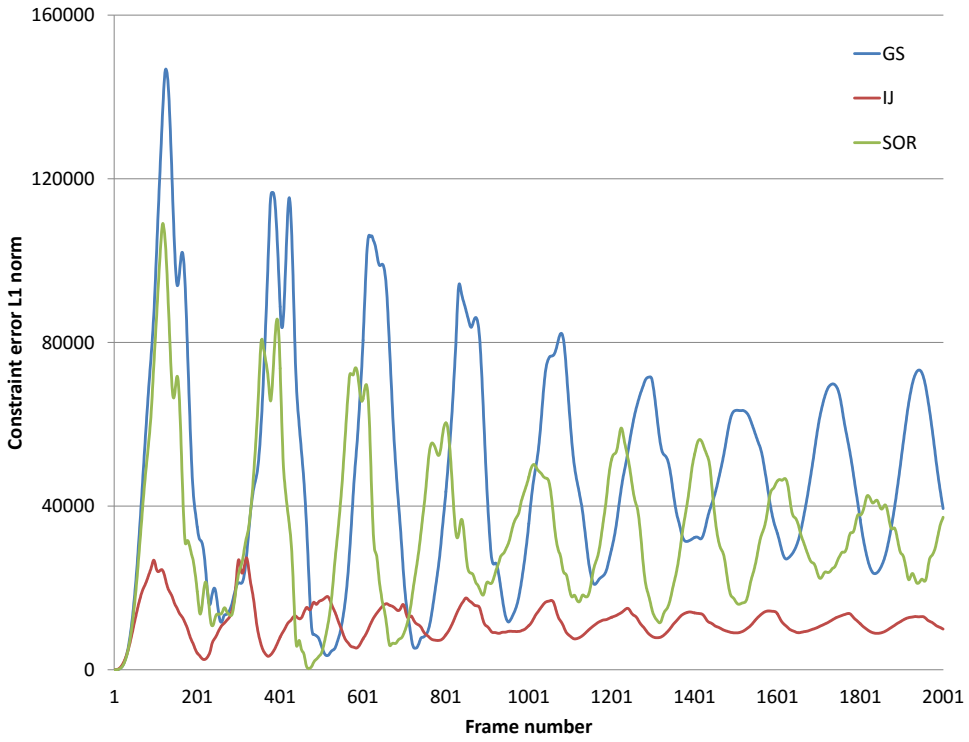


Figure 8.7: Plot of constraint error (L1 norm) for PBD cloth simulation with different solvers (frame number on the horizontal axis): Gauss-Seidel (blue), SOR (green) with $\omega = 1.2$, and improved Jacobi (red) with $\omega = 0.5$, $a = 1$ and $b = 0.6$.

Next we measured the positional errors for unilateral frictionless constraints by dropping 3000 particles in a box using a VTS method. As you can see from the results (Figure 8.8), the improved Jacobi method is not always more accurate than GS and we had to tweak it ($a = 2$) in order to get better results. Still, even without tweaking, our method behaves similarly to GS and at a similar cost (see Table 8.3).

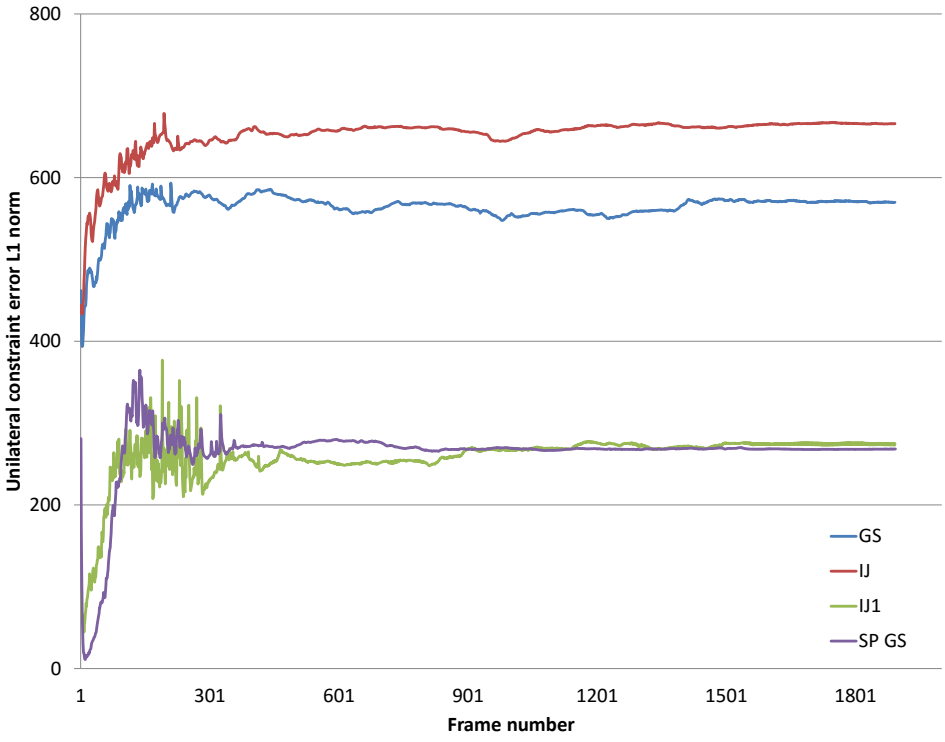


Figure 8.8: Plot of unilateral constraint error (L1 norm) for 3000 particles falling in a box (VTS, $\gamma = 0.5$): Gauss-Seidel (blue), improved Jacobi with $\omega = 0.5$, $a = 1$ and $b = 0.6$ (red), improved Jacobi with $a = 2$ (green), and Sequential Positions using GS (purple).

	Gauss-Seidel	Improved Jacobi
Cloth	39 ms	51 ms
Particles	5.2 ms	6.5 ms

Table 8.3: Frame time measurements made on a Intel Core i7 3770 CPU (single-threaded) for the two presented scenarios: hanging cloth (PBD) and falling particles (VTS).

We also tested heterogeneous mass values with large ratios between them and found that improved Jacobi handles them just as robustly as GS. Adding friction to the mix may introduce jitter but it can be alleviated by lowering ω. You can see the results of a falling particles simulation with inverse mass values between 0.01 and 1000 and friction ($\mu = 0.2$) in Figure 8.9.

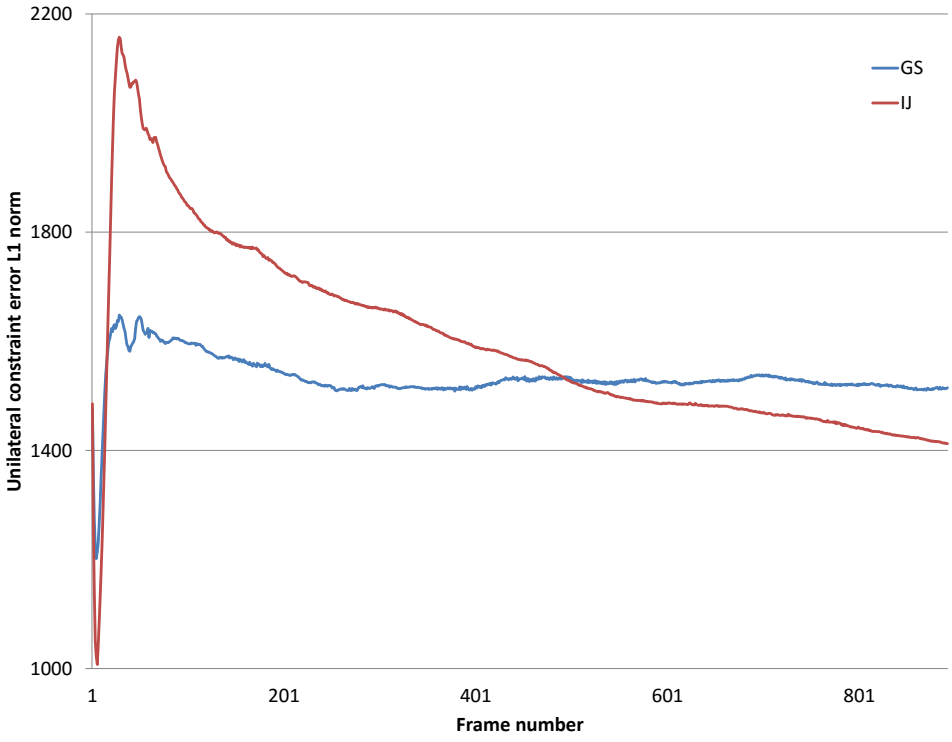


Figure 8.9: Plot of unilateral constraint error (L1 norm) for 3000 particles in a box with varying masses and friction: GS (blue), improved Jacobi (red) with $\omega = 0.4$, $a = 1$, $b = 0.6$.

In our falling particles experiments we used 15 iterations and one sub-step at 60 Hz, but the iteration count can be set even lower without breaking our method. For very low iteration numbers we recommend decreasing ω and b more in order to avoid jitter. Of course high velocities and small object sizes can put even GS in difficulty. In this situations increasing the number of iterations does not always work and we need to lower the time step, i.e. add more sub-steps.

In order to measure convergence per frame for both VTS and PBD/SP methods we switched to using the velocity error. We present results from frame 100 (Figure 8.10) for 1000 falling particles with friction (60 iterations per frame). Again improved Jacobi behaves very similarly to GS. Also our SP method converges better than VTS, but only for a high enough number

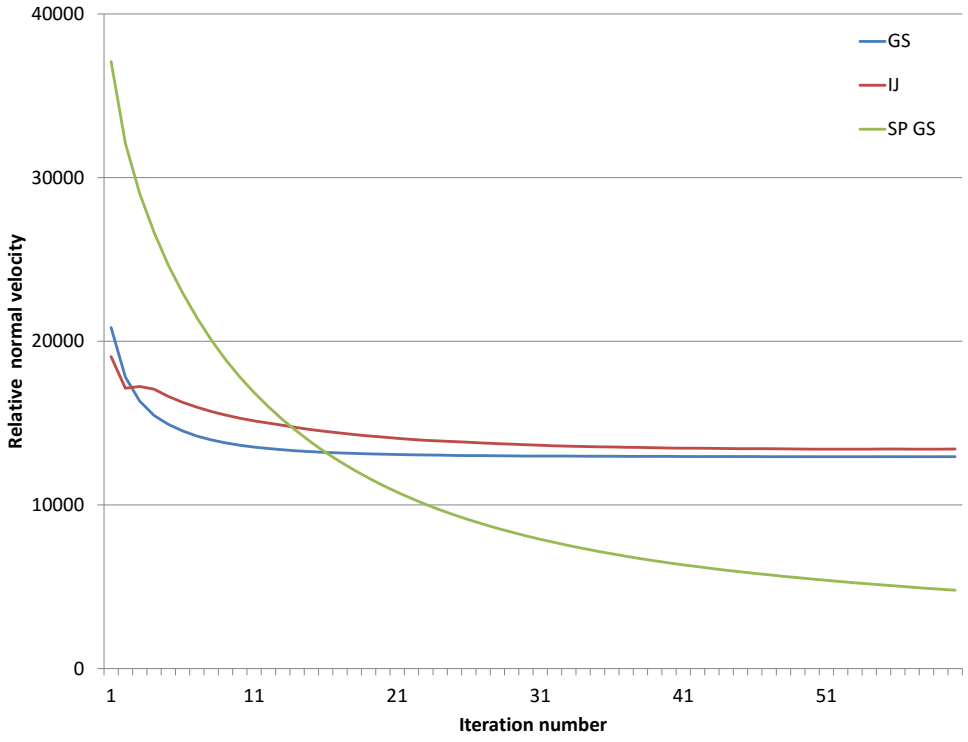


Figure 8.10: Plot of relative velocity along constraints relative to the number of iterations for 1000 particles falling in a box with contact and friction: Gauss-Seidel (blue), improved Jacobi (red) with $\omega = 0.5$, $a = 1$ and $b = 0.6$, and Sequential Positions using GS with $k_c = 1$ and $k_v = 0.1$ (green).

of iterations. You can see in Figure 8.8 that the positional error of the SP-GS method is also good but, for the visual aspect, impacts may require softening.

In [FM15a] we introduced the NCG solver (Section 4.3), soft constraints, better energy dissipation and conservation through the Newmark integrator (see also Section 6.5). Our most common test scenario was a piece of cloth hanging by two corners, falling from a horizontal or vertical position, with different parameters or tessellation. Given the multitude of methods used and the differences between them it is hard to find a metric that measures well the quality of the simulation. We opted to measure the total energy - kinetic and potential (gravitational and elastic) and no damping, and chart its evolution in time.

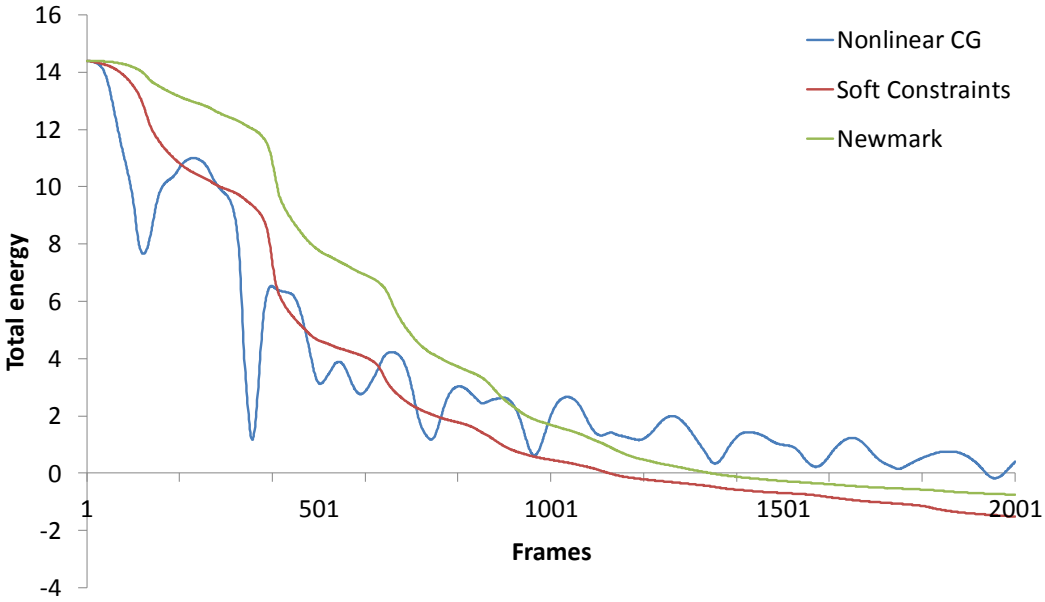


Figure 8.11: Total energy evolution in time for the simulation of a 10×10 rubber cloth ($\kappa = 2$ N/m, 25 iterations) using NCG implicit integration (blue), regularized PBD (red) and regularized energy preserving projection (green).

In Figure 8.11 you can see the NCG solver behaves well and has good convergence, but decays non-monotonically. The regularized PBD method is smoother, dissipates energy slower, but the Gauss-Seidel solver is less accurate. Energy preserving projection offers even slower energy decay while the higher energy line is due to the kinetic energy of the oscillation.

Realistic damping is quite hard to obtain. Reducing the velocity along constraint directions or explicitly applying damping forces many times results in increasing the system's oscillation. The method presented in [MHHR07] that blends between rigid and cloth motion remains a viable option, although what we really want to do is apply strong damping forces (like in a dash-pot) in a stable manner.

For our damping method (see Section 6.9) we measured the total energy minus the elastic potential in order to give a clearer picture of the velocity reduction (Figure 8.12). As you can see a damping factor of $\rho = 10h$ gives a significant energy dissipation compared to soft projection (or PBD just

as well). Reaching this level of dissipation so quickly is not possible using the method we compared against, i.e. reducing the relative velocity along the constraint direction (basically velocity projection). We used the energy preserving projector with a damping ratio $\rho = h$ in order to obtain as little artificial damping as possible while at same time damping the simulation just a bit less than PBD would normally do.

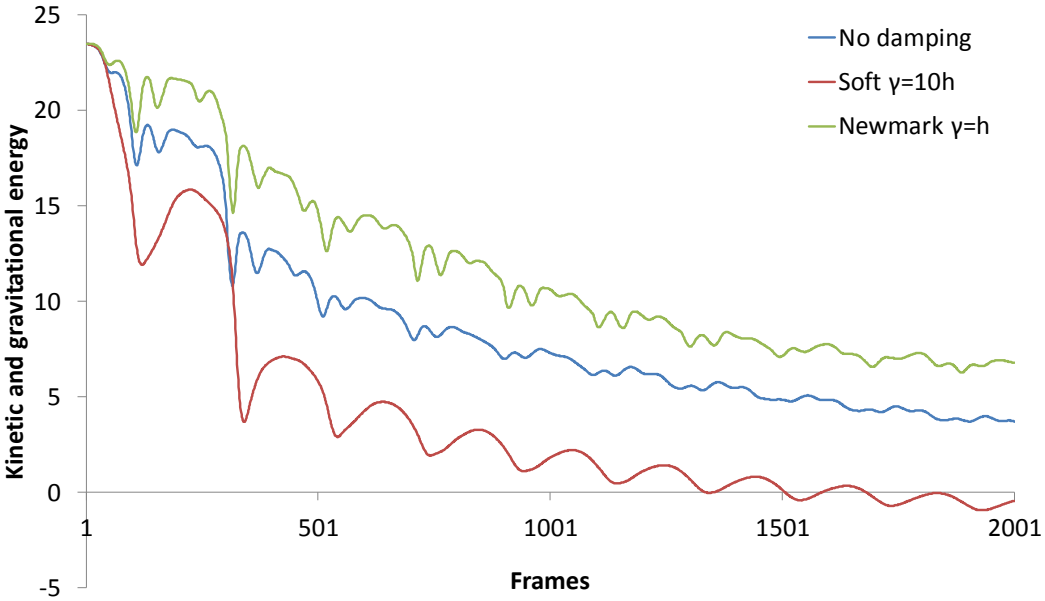


Figure 8.12: Damping response for the simulation of a 40×40 piece of cloth (κ = 2000 N/m, 25 iterations) using regularized PBD (blue), aggressive damping (red) and slightly damped energy preserving projection (green).

Collision detection was done using both Bullet [Cou10] and our own triangle mesh tests. We implemented our own code because we needed continuous collision detection when performing tests versus cloth or for self-collisions. We accelerated these tests using OpenMP loops and a variant of dynamic AABB trees. Collision detection is run right after the unconstrained position step when non-penetration constraints are sure to be violated. We do cloth-primitive and cloth-mesh intersection tests by testing all pairs of vertices and triangles or other primitives (e.g. sphere) in a similar fashion to [MHHR07] and [BFA02]. We also employed continuous collision detection techniques using the ideas presented in [Sta09] based on swept spheres. This was done

in order to catch tunneling artifacts although the contacts are still solved at the end of the time step, which is kept fixed.

Many of the simulations for this thesis were done in real-time inside our own OpenGL powered Windows application (see Figure 8.13). Others were done in an offline manner and then exported as Alembic geometry caches to Autodesk Maya and rendered using Pixar RenderMan. However, the simulator was written with real-time in mind and a lot of the scenarios ran at interactive rates, some even at 60 Hz. Generally we used a time step $h = 16$ ms, gravity $g = -9.8m/s^2$ and 10 to 50 iterations or more for our iterative solvers. For elastic bodies we used a Young's modulus $E = 0.5$ GPa and a Poisson ratio below 0.2. The masses of cloth and the soft bodies were raised up to around 10 kg in order interact smoothly with rigid bodies of unit mass or less.

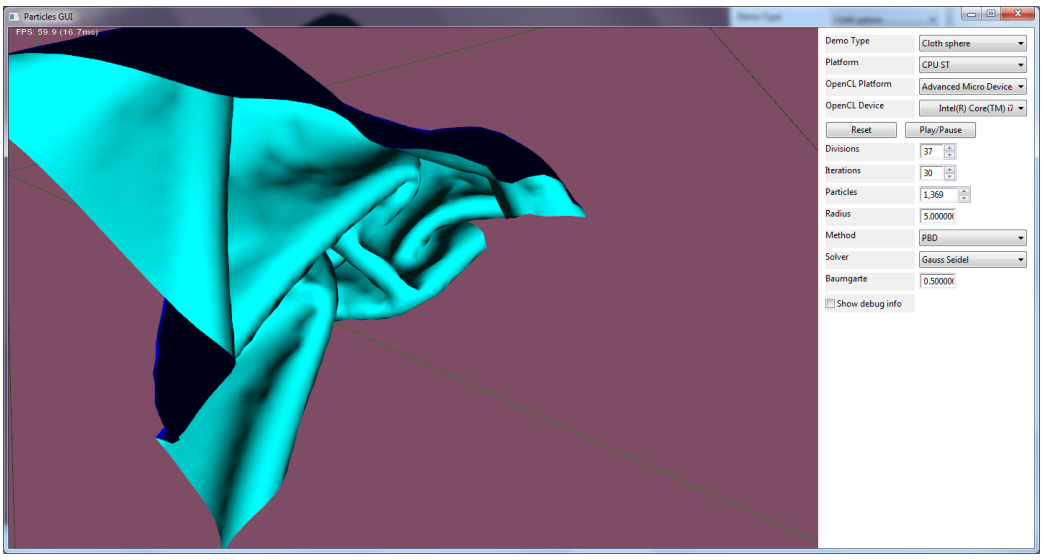


Figure 8.13: Windows application written in MS Visual C++ using OpenGL.

8.3 Mixing PBD and VTS

As not all types of bodies or steps in a constraint based simulator require nonlinear solvers, we looked at ways of combining PBD like methods with

velocity time stepping (VTS). For example PBD can be used as a post-stabilization step after VTS. This was already done in the past, but without updating the velocities too [CP03]. Also VTS can be used before PBD to soften the impacts. Thus we arrive at the conclusion that the two methods can be used sequentially to good results, similarly to the phase space projection approach in RATTLE.

Still one wonders whether a single solving step can address all the issues. VTS handles contacts well as it always tries to zero out relative velocities. PBD can be very violent as the relative velocities depend on the penetration which is brought near zero. Stewart referred to this as a random coefficient of restitution manifested by the nonlinear approach. On the other hand PBD handles bilateral constraints much better (especially when the direction is changing fast). Then there is also the issue of initial guesses: we found that not using the unconstrained step is often more stable for contacts. Possible solutions include: damped PBD, a nonlinear VTS scheme (more Newton-like iterations), limit the restitution velocity of PBD. Another thing we tried were hybrid schemes, where some constraints or subsystems are solved using either PBD or VTS. Unfortunately this did not work well in all cases (see Figure 8.14), but we are still investigating ways of coupling the two methods. One outcome would be using VTS rigids or VTS-FEM which is faster in conjunction with PBD cloth or hair.

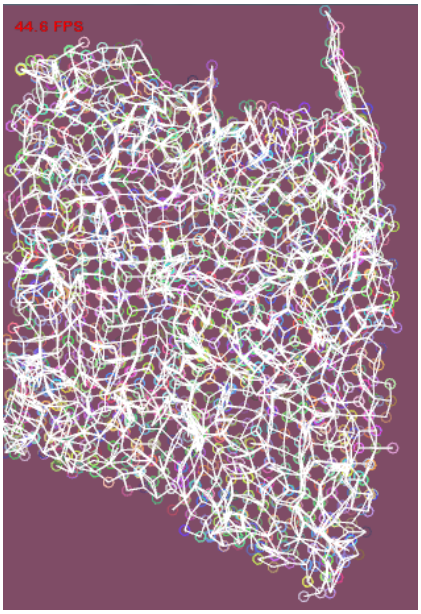


Figure 8.14: Hybrid method - VTS for contacts after PBD for links

Chapter 9

Conclusions and future work

9.1 Conclusions

We did not answer all the questions or fulfill all the goals we set in the introduction. But we did clarify a lot of aspects during this doctoral research. For instance, we know now that position based dynamics is truly a physically correct method and there is no cheating involved. It is also deeply connected to velocity time stepping methods used in rigid body simulation: VTS is just a linearization of PBD. We also found out that both PBD and VTS can be expressed as optimization problems and new solving strategies can be used. Even in the case of frictional contact we no longer need to express the problem solely as a LCP but we can turn it into a convex minimization. And this allows us a unifying view of all constraints and a general formulation of PBD as a fully implicit and nonlinear projection scheme.

We have also learned the relationship between Lagrange multiplier methods and penalty methods. If one finds ideal constraints too restrictive she can still use force based constitutive laws under the same formulation and solving approach of PBD and VTS. This is done through softening the constraints (same as adding compliance or regularization) which is a completely physical process. You can use it to turn a rigid link into a spring or a tetrahedron into a finite element. We have shown proof for these arguments throughout the thesis and also made the important point that the residual of the

iterative solvers is what really introduces elasticity into constrained systems. In conclusion, constraint based solvers are very good at handling very stiff systems, with the exact solution corresponding to infinite stiffness. Otherwise, when regularized they correspond to the real physical models with given constitutive parameters. So PBD can really be physically correct from this viewpoint, and is actually better than VTS which is linear, semi-implicit and has stability issues.

The same reasoning can be applied to contact: the complementarity approach is just the stiffness limit of the penalty method. And the setting of nonsmooth dynamics allows for treating impulsive forces like impact and friction ones in a simple time stepping manner. We did our best in this thesis to prove that the apparatus developed for nonsmooth VTS methods does also apply for PBD. And our final conclusion was that PBD can be expressed as a series of stabilized VTS steps or a fixed point iteration designed to solve the original nonlinear problem.

All these theoretical findings gave us confirmation to build a unified simulation implementation. This was not essentially new but it now has an underlying mathematical formulation to build upon. We can now simulate rigid bodies, particles, cloth, soft bodies and possibly many others in the future using one single constraint solver. We stress the fact two way coupling is achieved out of the box in this way and there is no need for co-simulation, i.e. mixing penalty solvers with constraint based ones. One can still add forces to the mix (even in an implicit way) but the beauty of our solving approach is that it is matrix free and easy to grasp and implement.

The author is very happy to have achieved all these things and most importantly to have understood all these phenomena and numerical methods better. Also he wonders why all these things were not clear from the start. Even though some authors do seem to have hinted in these directions, the real challenge was in selecting the signal from the noise. There is still much to learn and work on ahead, but we are content of knowing now how the puzzle pieces fit together. We hope that the reader has also had a good experience reading this thesis and that she also learned new things about simulation methods, how they relate to each other and how they can be

tackled numerically.

9.2 Contributions

We will now give a brief list of the original contributions presented in this thesis together with a reference to the section where they are presented and the article where they were published (where applicable):

- a nonlinear conjugate gradient (NCG) solver for cloth simulation - presented in Section 4.3 and published in [FM15a];
- a general formulation of position based dynamics as a minimization problem with nonlinear constraints (Section 6.4) equivalent with implicit Euler integration; we first used this formulation in an SQP solving approach in terms of dual variables and published it in [FM14b];
- a physical foundation for the minimization formulation based on variational principles of mechanics and the Newmark integrator (Section 6.5);
- a nonlinear minimum residual and a conjugate residuals (CR) solver for the position based dynamics optimization problem (Section 6.6.2) also published in [FM14b];
- an improved Jacobi scheme (with two variants) based on an extra momentum term that has better convergence than standard Jacobi, is comparable to Gauss-Seidel and can be parallelized (Section 6.6.3);
- an equivalence result between the implicit integration of elastic potentials and position based dynamics with soft constraints (or regularization - Section 6.8) published in [FM15a];
- an explanation to why conjugate residuals work better than conjugate gradients for solving constraints based on the transformation from primal to dual variables (Section 6.8 and [FM15a]);

- an alternative way of damping constraint based simulations (Section 6.9), also derived in a different manner in [FM15a] where we also introduced the energy preserving Newmark projection (Section 6.5);
- an accurate finite element method expressed in terms of constraint solving (Section 6.11) based on two aforementioned equivalences: 1. position based dynamics is equivalent to implicit integration and 2. elastic potentials are equivalent to regularized constraints (see also [FM15a] for an application in cloth simulation);
- an accurate frictional contact model for position based dynamics stemming from nonsmooth dynamics together with a convergence proof (Chapter 7);
- a rigid body simulator using position based dynamics and the above frictional contact model (Section 7.7).
- two way coupling between rigid bodies and deformable bodies (including particles and cloth) obtained for free by modeling all interactions as constraints and running them in the same solver, e.g. cloth links, FEM constraints, contacts with friction etc. (Chapter 8).

9.3 Future work

There are many topics that we did not get enough time to study, investigate, understand or implement. In fact they are so many that we could fill several pages talking about them. Instead we will try to be as brief as possible. This section is about those topics that we would like to continue working on after the PhD is over.

Given that we studied contact last and we ran out of time before trying out everything we had in mind it is natural that we would like to continue that work (it would fill another thesis probably). We would have liked to have a benchmark system made out of nonsmooth dynamics methods not only from Stewart, Anițescu, Negruț, Tasora but also others, e.g. the sweeping process, the nonsmooth contact dynamics (NSCD) method, variational

contact integrators and so on. This is not only for comparison reasons but also to better understand the DVI formulation and subtleties about its solutions and proofs. We are still in need of a sensitivity result for conic QPs. We also want to implement at least one variant of the Newton approaches to the complementarity problem; the same applies for saddle point problem approaches. Even for the LCP and CCP discretization we still plan to test multigrid methods and other iterative solvers: spectral projected gradient methods (e.g. Barzilai-Borwein), accelerated gradient methods (e.g. Nesterov) and other improvements to relaxation schemes. We need to spend more time on our position based rigid body simulator and improve stacking. Besides these we think the following items are still open problems not only for us but the whole field of nonsmooth contact dynamics:

- friction model with cylindrical projection,
- methods that guarantee the coefficient of restitution,
- artifacts from convexification,
- sources of jitter and solutions to alleviate it at low iteration count.

Regarding bilateral constraints we would like to spend some time on the subject of hair simulation. We also wanted to extend our FEM support to fracture, invertible elements, model reduction and other elasticity models, e.g. co-rotational or neo-Hookean. We hope that our soft constraints approach extends to nonlinear springs too. We need to study more the relationship between the number of iterations and the observed stiffness and behavior. We want to find out how badly this dependence affects the quality of the simulation and the appeal of our approach in comparison to other methods, e.g. implicit integration, projective dynamics. We definitely need to clarify if PBD and VTS can be mixed together and how.

One aspect we would have liked to know sooner is that constrained systems can be solved in either primal variables (velocities) or dual ones (constraint forces). This is a mind opener as many more algorithms can be used in either of the forms and it makes the equivalence to implicit springs more

obvious. If we were to do our research all over again we would start from here. We would also use more off the shelf optimization solvers like Mosek, PATH or the ones from MATLAB.

Simulation-wise we want to spend more time on hair and fluids. We would like to approach the latter in a constraint based approach. This has already been done with constraint fluids in a Lagrangian view or optimization formulations in the Eulerian one. We also did not have time to build real complex simulation scenarios to show off our coupling and friction results and we intend to address this in the future.

For the implementation part there is of course a lot of work to do too. Our biggest remaining task is to implement a full unified physics pipeline on the GPU (and replace OpenCL by CUDA). Collision detection also needs a lot of work on many aspects:

- self-collision handling and CCD,
- broadphase and midphase acceleration structures (e.g. BVH),
- triangle mesh penetration tests and contact normals generation,
- contact manifold reduction,
- overall quality of contacts (e.g. spurious normals).

Bibliography

[AB08] Vincent Acary and Bernard Brogliato. *Numerical Methods for Nonsmooth Dynamical Systems: Applications in Mechanics and Electronics*. Springer Science & Business Media, 2008.

[ACLM11] Vincent Acary, Florent Cadoux, Claude Lemaréchal, and Jérôme Malick. A Formulation of the Linear Discrete Coulomb Friction Problem via Convex Optimization. *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik*, 91(2):155–175, 2011.

[ACPR95] Uri M. Ascher, Hongsheng Chin, Linda R. Petzold, and Sebastian Reich. Stabilization of Constrained Mechanical Systems with Daes and Invariant Manifolds. *Journal of Structural Mechanics*, 23(2):135–157, 1995.

[AH04a] Mihai Anițescu and Gary D. Hart. A Constraint-Stabilized Time-Stepping Approach for Rigid Multibody Dynamics with Joints, Contact and Friction. *International Journal for Numerical Methods in Engineering*, 60(14):2335–2371, 2004.

[AH04b] Mihai Anițescu and Gary D. Hart. A Fixed-Point Iteration Approach for Multibody Dynamics with Contact and Small Friction. *Mathematical Programming*, 101(1):3–32, 2004.

[And83] Hans C. Andersen. Rattle: A "Velocity" Version of the Shake Algorithm for Molecular Dynamics Calculations. *Journal of Computational Physics*, 52(1):24–34, 1983.

[Ani06] Mihai Anițescu. Optimization-Based Simulation of Nonsmooth Rigid Multibody Dynamics. *Mathematical Programming*, 105(1):113–143, 2006.

[AO11] Iván Alduán and Miguel A. Otaduy. SPH Granular Flow with Friction and Cohesion. In *ACM SIGGRAPH 2011 Papers*, SIGGRAPH '11, New York, NY, USA, 2011. ACM.

[AP97] Mihai Anițescu and Florian A. Potra. Formulating Dynamic Multi-Rigid-Body Contact Problems with Friction as Solvable Linear Complementarity Problems. *Nonlinear Dynamics*, 14(3):231–247, 1997.

[AP02] Mihai Anițescu and Florian A. Potra. A Time-Stepping Method for Stiff Multibody Dynamics with Contact and Friction. *International Journal for Numerical Methods in Engineering*, 55(7):753–784, 2002.

[Arn13] Vladimir Igorevich Arnol'd. *Mathematical Methods of Classical Mechanics*, volume 60. Springer Science & Business Media, 2013.

[ATO09] Iván Alduán, Angel Tena, and Miguel A. Otaduy. Simulation of High-Resolution Granular Media. In *Proc. of Congreso Español de Informática Gráfica*, volume 1, 2009.

[BAC+06] Florence Bertails, Basile Audoly, Marie-Paule Cani, Bernard Querleux, Frédéric Leroy, and Jean-Luc Lévêque. Super-Helices for Predicting the Dynamics of Natural Hair. *ACM Transactions on Graphics (TOG)*, 25(3):1180–1187, 2006.

[Bar94] David Baraff. Fast Contact Force Computation for Nonpenetrating Rigid Bodies. In *ACM SIGGRAPH 1994 Papers*, pages 23–34, 1994.

[Bar97] David Baraff. Physically Based Modeling: Rigid Body Simulation. *SIGGRAPH Course Notes, ACM SIGGRAPH*, 2(1):2–1, 1997.

[Bat06] Klaus-Jürgen Bathe. *Finite Element Procedures*. Prentice Hall, Pearson Education, Inc., 2006.

[Bau72] Joachim Baumgarte. Stabilization of Constraints and Integrals of Motion in Dynamical Systems. *Computer methods in applied mechanics and engineering*, 1(1):1–16, 1972.

[BBD09] Jan Bender, Daniel Bayer, and Raphael Diziol. Dynamic Simulation of Inextensible Cloth. *IADIS International Journal on Computer Science and Information Systems*, 4(2):86–102, 2009.

[BBH08] Derek Bradley, Tamy Boubekeur, and Wolfgang Heidrich. Accurate Multi-View Reconstruction Using Robust Binocular Stereo and Surface Meshing. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE, 2008.

[BCP96] Kathryn Eleda Brenan, Stephen L. Campbell, and Linda Ruth Petzold. *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*, volume 14. Siam, 1996.

[BDCDA11] Florence Bertails-Descoubes, Florent Cadoux, Gilles Daviet, and Vincent Acary. A Nonsmooth Newton solver for Capturing Exact Coulomb Friction in Fiber Assemblies. *ACM Transactions on Graphics (TOG)*, 30(1):6, 2011.

[BETC14] Jan Bender, Kenny Erleben, Jeff Trinkle, and Erwin Coumans. Interactive Simulation of Rigid Body Dynamics in Computer Graphics. *Computer Graphics Forum*, 33(1):246–270, 2014.

[BFA02] Robert Bridson, Ronald Fedkiw, and John Anderson. Robust Treatment of Collisions, Contact and Friction for Cloth Animation. In *ACM SIGGRAPH 2002 Papers*, pages 594–603, 2002.

[BHW94] David E. Breen, Donald H. House, and Michael J. Wozny. Predicting the Drape of Woven Cloth Using Interacting Particles. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 365–372. ACM, 1994.

[BKCW14] Jan Bender, Dan Koschier, Patrick Charrier, and Daniel Weber. Position-Based Simulation of Continuous Materials. *Computers & Graphics*, 44:1–10, 2014.

[BKLS95] Eric Barth, Krzysztof Kuczera, Benedict Leimkuhler, and Robert D. Skeel. Algorithms for Constrained Molecular Dynamics. *J. Comp. Chem*, 16:1192–1209, 1995.

[BLS12] Kenneth Bodin, Claude Lacoursière, and Martin Servin. Constraint fluids. *Visualization and Computer Graphics, IEEE Transactions on*, 18(3):516–526, 2012.

[BMF03] Robert Bridson, Sebastian Marino, and Ronald Fedkiw. Simulation of Clothing with Folds and Wrinkles. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 28–36, 2003.

[BML+14] Sofien Bouaziz, Sebastian Martin, Tiantian Liu, Ladislav Kavan, and Mark Pauly. Projective Dynamics: Fusing Constraint Projections for Fast Simulation. *ACM Trans. Graph.*, 33(4):154:1–154:11, 2014.

[BMOT13] Jan Bender, Matthias Müller, Miguel A. Otaduy, and Matthias Teschner. Position-based Methods for the Simulation of Solid Objects in Computer Graphics. In *EUROGRAPHICS 2013 State of the Art Reports*. Eurographics Association, 2013.

[BP97] Marshall Wayne Bern and Paul E. Plassmann. *Mesh Generation*. Pennsylvania State University, Department of Computer Science and Engineering, College of Engineering, 1997.

[BPS+08] Derek Bradley, Tiberiu Popa, Alla Sheffer, Wolfgang Heidrich, and Tamy Boubekeur. Markerless Garment Capture. In *ACM Transactions on Graphics (TOG)*, volume 27, page 99. ACM, 2008.

[Bri14] Robert Bridson. Animating Cloth with Coupled Contact (a Quick and Dirty Approach). *Computational Contact Mechanics: Advances and Frontiers in Modeling Contact*, 2014.

[Bri15] Robert Bridson. *Fluid Simulation for Computer Graphics*. CRC Press, 2015.

[Bro96] Bernard Brogliato. Nonsmooth Impact Mechanics (Models, Dynamics and Control). *Lecture notes in control and information sciences*, 1996.

[BTH+03] Kiran S. Bhat, Christopher D. Twigg, Jessica K. Hodgins, Pradeep K. Khosla, Zoran Popović, and Steven M. Seitz. Estimating Cloth Simulation Parameters from Video. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 37–51. Eurographics Association, 2003.

[BV04] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[BW97] Javier Bonet and Richard D. Wood. *Nonlinear Continuum Mechanics for Finite Element Analysis*. Cambridge University Press, 1997.

[BW98] David Baraff and Andrew Witkin. Large Steps in Cloth Simulation. In *ACM SIGGRAPH 1998 Papers*, pages 43–54, 1998.

[BWH+06] Miklos Bergou, Max Wardetzky, David Harmon, Denis Zorin, and Eitan Grinspun. A Quadratic Bending Model for Inextensible Surfaces. In *Symposium on Geometry Processing*, pages 227–230, 2006.

[BYM05] Nathan Bell, Yizhou Yu, and Peter J. Mucha. Particle-based Simulation of Granular Materials. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '05, pages 77–86, New York, NY, USA, 2005. ACM.

[BZX14] Jernej Barbic, Yili Zhao, and Hongyi Xu. Implicit Multibody Penalty-based Distributed Contact. *IEEE Transactions on Visualization and Computer Graphics*, 99(PrePrints):1, 2014.

[Cat05] Erin Catto. Iterative Dynamics with Temporal Coherence. *Game Developer Conference*, January 2005.

[Cat10] Erin Catto. Soft Constraints: Reinventing the Spring. *Game Developer Conference*, 2010.

[CC13] Teodor Cioacă and Horea Cărămizaru. On the Impact of Explicit or Semi-Implicit Integration Methods Over the Stability of Real-Time Numerical Simulations. *arXiv preprint arXiv:1311.5018*, 2013.

[CK02] Kwang-Jin Choi and Hyeong-Seok Ko. Stable but Responsive Cloth. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '02, pages 604–611, 2002.

[Cou10] Erwin Coumans. Bullet Physics Engine. *Open Source Software: http://bulletphysics.org*, 2010.

[Cou12] Erwin Coumans. Destruction. In *Game Developers Conference Proceedings*. CMP Media, Inc., 2012.

[Cou14] Erwin Coumans. Exploring MLCP Solvers and Featherstone. *GDC*, 2014.

[CP03] Michael B. Cline and Dinesh K. Pai. Post-Stabilization for Rigid Body Simulation with Contact and Constraints. In

Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on, volume 3, pages 3744–3751. IEEE, 2003.

[CPS92] Richard W. Cottle, Jong-Shi Pang, and Richard E. Stone. *The Linear Complementarity Problem.* Classics in Applied Mathematics. Society for Industrial and Applied Mathematics (SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104), 1992.

[DCB14] Crispin Deul, Patrick Charrier, and Jan Bender. Position-Based Rigid Body Dynamics. In *Proceedings of the 27th International Conference on Computer Animation and Social Agents*, May 2014.

[DSF98] Géry De Saxcé and Zhi-Qiang Feng. The Bipotential Method: A Constructive Approach to Design the Complete Contact Law with Friction and Improved Numerical Algorithms. *Mathematical and Computer Modeling*, 28(4):225–245, 1998.

[DTE+04] Andreas Divivier, Rainer Trieb, Aea Ebert, Hans Hagen, Clemens Gross, Arnulph Fuhrmann, Volker Luckas, et al. Virtual Try-On Topics in Realistic, Individualized Dressing in Virtual Reality. 2004.

[Dug57] René Dugas. *A History of Mechanics.* JR Maddox. London, 1957.

[EB08] Elliot English and Robert Bridson. Animating Developable Surfaces Using Nonconforming Elements. *ACM Trans. Graph.*, 27(3):66:1–66:5, 2008.

[EEH00] Bernhard Eberhardt, Olaf Etzmuß, and Michael Hauth. Implicit-Explicit Schemes for Fast Animation with Particle Systems. In *In Eurographics Computer Animation and Simulation Workshop*, pages 137–151, 2000.

[Eri04] Christer Ericson. *Real-Time Collision Detection*. CRC Press, 2004.

[Erl05] Kenny Erleben. *Physics-based Animation*. Charles River Media, 2005.

[Erl07] Kenny Erleben. Velocity-based Shock Propagation for Multibody Dynamics Animation. *ACM Trans. Graph.*, 26, 2007.

[Fau99] François Faure. *Interactive Solid Animation Using Linearized Displacement Constraints*. Springer, 1999.

[Fea14] Roy Featherstone. *Rigid Body Dynamics Algorithms*. Springer, 2014.

[Fin09] J. Michael Finn. *Classical Mechanics*. Jones & Bartlett Publishers, 2009.

[FM14a] Mihai Francu and Florica Moldoveanu. An Improved Jacobi Solver for Particle Simulation. In *VRIPHYS 14 - 11th Workshop on Virtual Reality Interactions and Physical Simulations*, pages 125–134, 2014.

[FM14b] Mihai Frâncu and Florica Moldoveanu. Minimum Residual Methods for Cloth Simulation. In *System Theory, Control and Computing (ICSTCC), 2014 18th International Conference*, pages 550–555, Oct 2014.

[FM15a] Mihai Frâncu and Florica Moldoveanu. Cloth simulation using soft constraints. *Journal of WSCG*, 2015.

[FM15b] Mihai Frâncu and Florica Moldoveanu. Virtual Try On Systems for Clothes: Issues and Solutions. *Scientific Bulletin*, 2015.

[FMOW03] Răzvan C. Fetecău, Jerrold E. Marsden, Michael Ortiz, and Matthew West. Nonsmooth Lagrangian Mechanics and Variational Collision Integrators. *SIAM Journal on Applied Dynamical Systems*, 2(3):381–416, 2003.

[GBF03] Eran Guendelman, Robert Bridson, and Ronald Fedkiw. Nonconvex Rigid Bodies with Stacking. In *ACM SIGGRAPH 2003 Papers*, SIGGRAPH '03, pages 871–878, New York, NY, USA, 2003. ACM.

[GHF+07] Rony Goldenthal, David Harmon, Raanan Fattal, Michel Bercovier, and Eitan Grinspun. Efficient Simulation of Inextensible Cloth. In *ACM SIGGRAPH 2007 Papers*, 2007.

[Gol10] Adrian R. Goldenthal. *Implicit Treatment of Constraints for Cloth Simulation*. PhD thesis, 2010.

[GPS02] Herbert Goldstein, Charles P. Poole, and John L. Safko. *Classical Mechanics*. Addison Wesley, 2002.

[GS02] Gianni Gilardi and Inna Sharf. Literature Survey of Contact Dynamics Modelling. *Mechanism and Machine Theory*, 37(10):1213–1239, 2002.

[Had06] Sunil Hadap. Oriented Strands: Dynamics of Stiff Multi-Body System. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 91–100. Eurographics Association, 2006.

[Han06] Andrew J. Hanson. *Visualizing Quaternions (The Morgan Kaufmann Series in Interactive 3D Technology)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2006.

[Har07] Takahiro Harada. Real-Time Rigid Body Simulation on GPUs. In Hubert Nguyen, editor, *GPU Gems 3*, chapter 29. Addison Wesley Professional, August 2007.

[HATN12] Toby Heyn, Mihai Anițescu, Alessando Tasora, and Dan Negruț. Using Krylov Subspace and Spectral Methods for Solving Complementarity Problems in Many-Body Contact Dynamics Simulation. *International Journal for Numerical Methods in Engineering*, 2012.

[Hau05] Michael Hauth. Numerical Techniques for Cloth Simulation. *system (figure 2 (a)*, 15:3, 2005.

[HCJ+05] Min Hong, Min-Hyung Choi, Sunhwa Jung, Samuel Welch, and John Trapp. Effective Constrained Dynamic Simulation Using Implicit Constraint Enforcement. In *Robotics and Automation, ICRA 2005*, pages 4520–4525, 2005.

[HE01] Michael Hauth and Olaf Etzmuss. A High Performance Solver for the Animation of Deformable Objects Using Advanced Numerical Methods. In *Eurographics 2001*, volume 20, pages 319–328, 2001.

[HH12] Dongsoo Han and Takahiro Harada. Real-Time Hair Simulation with Efficient Hair Style Preservation. 2012.

[HLR89] Ernst Hairer, Christian Lubich, and Michel Roche. The Numerical Solution of Differential-Algebraic Systems by Runge-Kutta Methods. 1989.

[HLW03] Ernst Hairer, Christian Lubich, and Gerhard Wanner. Geometric Numerical Integration Illustrated by the Störmer–Verlet Method. *Acta Numerica*, 12:399–450, 2003.

[HLW06] Ernst Hairer, Christian Lubich, and Gerhard Wanner. *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations*, volume 31. Springer Science & Business Media, 2006.

[HNW87] Ernst Hairer, Syvert Paul Norsett, and Gerhard Wanner. Solving ordinary differential equation I: nonstiff problems. *Springer Ser. in Comput. Math*, 8, 1987.

[How11] Lee Howes. Cloth Simulation in the Bullet Physics SDK. In Aaftab Munshi, Benedict R. Gaster, Timothy G. Mattson, James Func, and Dan Ginsburg, editors, *OpenCL Programming Guide*, chapter 17, page 425–448. Addison Wesley, 06 2011.

[HS07] Paul Hellard and Jos Stam. Stam on Maya's nCloth. *CGSociety*, May 2007.

[HY90] Edward J. Haug and Jeng Yen. Generalized Coordinate Partitioning Methods for Numerical Integration of Differential-Algebraic Equations of Dynamics. In *Real-time Integration Methods for Mechanical System Simulation*, pages 97–114. Springer, 1990.

[HZ06] William W. Hager and Hongchao Zhang. A survey of Nonlinear Conjugate Gradient Methods. *Pacific Journal of Optimization*, 2(1):35–58, 2006.

[IWT12] Markus Ihmsen, Arthur Wahl, and Matthias Teschner. High-Resolution Simulation of Granular Material with SPH. In *Workshop on Virtual Reality Interaction and Physical Simulation*, pages 53–60. The Eurographics Association, 2012.

[JAJ98] Franck Jourdan, Pierre Alart, and Michel Jean. A Gauss-Seidel like Algorithm to Solve Frictional Contact Problems. *Computer Methods in Applied Mechanics and Engineering*, 155(1):31–47, 1998.

[Jak01] Thomas Jakobsen. Advanced Character Physics. In *Game Developers Conference Proceedings*, pages 383–401, 2001.

[Jea99] Michel Jean. The Non-Smooth Contact Dynamics Method. *Computer Methods in Applied Mechanics and Engineering*, 177(3):235–257, 1999.

[JNB96] Heinrich M. Jaeger, Sidney R. Nagel, and Robert P. Behringer. Granular Solids, Liquids, and Gases. *Rev. Mod. Phys.*, 68:1259–1273, Oct 1996.

[KGBS11] Ladislav Kavan, Dan Gerszewski, Adam W. Bargteil, and Peter-Pike Sloan. Physics-Inspired Upsampling for Cloth Sim-

ulation in Games. *ACM Transactions on Graphics (TOG)*, 30(4):93, 2011.

[KKN+13] Doyub Kim, Woojong Koh, Rahul Narain, Kayvon Fatahalian, Adrien Treuille, and James F. O'Brien. Near-Exhaustive Precomputation of Secondary Cloth Effects. *ACM Transactions on Graphics (TOG)*, 32(4):87, 2013.

[KMOW99] Couro Kane, Jerrold E. Marsden, Michael Ortiz, and Matthew West. Variational Integrators and the Newmark Algorithm for Conservative and Dissipative Mechanical Systems. *Internat. J. Numer. Methods Engrg*, 49:1295–1325, 1999.

[KNE10] Micky Kelager, Sarah Niebe, and Kenny Erleben. A Triangle Bending Constraint Model for Position-Based Dynamics. *VRIPHYS*, 10:31–37, 2010.

[KNM10] Shoji Kunitomo, Shinsuke Nakamura, and Shigeo Morishima. Optimization of Cloth Simulation Parameters by Considering Static and Dynamic Features. In *ACM SIGGRAPH 2010 Posters*, page 15. ACM, 2010.

[Kny10] Anton Knyazyev. Ropes as Constraints. In *Game Physics Pearls*, pages 179–193. AK Peters/CRC Press, 2010.

[KP12] Danny M. Kaufman and Dinesh K. Pai. Geometric Numerical Integration of Inequality Constrained, Nonsmooth Hamiltonian systems. *SIAM Journal on Scientific Computing*, 34(5):A2670–A2703, 2012.

[KPGF07] Blazej Kubiak, Nico Pietroni, Fabio Ganovelli, and Marco Fratarcangeli. A Robust Method for Real-Time Thread Simulation. In *Proceedings of the 2007 ACM symposium on Virtual reality software and technology*, pages 85–88. ACM, 2007.

[KROM99] Couro Kane, E.A. Repetto, Michael Ortiz, and Jerrold E. Marsden. Finite Element Analysis of Nonsmooth Contact. *Computer*

Methods in Applied Mechanics and Engineering, 180(1):1–26, 1999.

[KSJP08] Danny M. Kaufman, Shinjiro Sueda, Doug L. James, and Dinesh K. Pai. Staggered Projections for Frictional Contact in Multibody Systems. *ACM Transactions on Graphics (SIGGRAPH Asia 2008)*, 27(5):164:1–164:11, 2008.

[KTS+14] Danny M. Kaufman, Rasmus Tamstorf, Breannan Smith, Jean-Marie Aubry, and Eitan Grinspun. Adaptive Nonlinearity for Collisions in Complex Rod Assemblies. *ACM Trans. Graph.*, 33(4):123:1–123:12, July 2014.

[KYT+06a] Liliya Kharevych, Weiwei Yang, Yiying Tong, Eva Kanso, Jerrold E Marsden, Peter Schröder, and Matthieu Desbrun. Geometric, Variational Entegrators for Computer Animation. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 43–51. Eurographics Association, 2006.

[KYT+06b] Liliya Kharevych, Weiwei Yang, Yiying Tong, Eva Kanso, Jerrold E. Marsden, Peter Schröder, and Matthieu Desbrun. Geometric, Variational Integrators for Computer Animation. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 43–51. Eurographics Association, 2006.

[Lac03] Claude Lacoursière. Splitting Methods for Dry Frictional Contact Problems in Rigid Multibody Systems: Preliminary Performance Results. In *The Annual SIGRAD Conference. Special Theme – Real-Time Simulations. Conference Proceedings from SIGRAD2003*, 2003.

[Lac07] Claude Lacoursière. *Ghosts and Machines: Regularized Variational Methods for Interactive Simulations of Multibodies with*

Dry Frictional Contacts. PhD thesis, Umeå University, Computing Science, 2007.

[Lan70] Cornelius Lanczos. *The Variational Principles of Mechanics*. Dover Publications, 1970.

[LBOK13] Tiantian Liu, Adam W. Bargteil, James F. O'Brien, and Ladislav Kavan. Fast Simulation of Mass-spring Systems. *ACM Trans. Graph.*, 32(6):214:1–214:7, 2013.

[Lew03] Adrián Lew. *Variational Time Integrators in Computational Solid Mechanics*. PhD thesis, California Institute of Technology, 2003.

[LL60] Lev D. Landau and Evgeny M. Lifshitz. *Mechanics*. 1960.

[LR04] Benedict Leimkuhler and Sebastian Reich. *Simulating Hamiltonian Dynamics*, volume 14. Cambridge University Press, 2004.

[LSB10] Claude Lacoursière, Martin Servin, and Anders Backman. Fast and Stable Simulation of Granular Matter and Machines. 2010.

[LY84] David G. Luenberger and Yinyu Ye. *Linear and Nonlinear Programming*, volume 2. Springer, 1984.

[LZY10] Yong-Jin Liu, Dong-Liang Zhang, and Matthew Ming-Fai Yuen. A Survey on CAD Methods in 3D Garment Design. *Computers in Industry*, 61(6):576–593, 2010.

[Mar03] F. Landis Markley. Attitude Error Representations for Kalman Filtering. *Journal of Guidance, Control, and Dynamics*, 26(2):311–317, 2003.

[Maz16] Hammad Mazhar. *Multi-Physics Computational Dynmics Using Complementarity and Hybrid Lagrangian-Eeulerian Methods*. PhD thesis, 2016.

[MBPV11] Lorenzo Mariti, Nicola P. Belfiore, Ettore Pennestrì, and Pier P. Valentini. Comparison of Solution Strategies for Multibody Dynamics Equations. *International Journal for Numerical Methods in Engineering*, 88(7):637–656, 2011.

[MBT+12] Eder Miguel, Derek Bradley, Bernhard Thomaszewski, Bernd Bickel, Wojciech Matusik, Miguel A. Otaduy, and Steve Marschner. Data-Driven Estimation of Cloth Simulation Models. In *Computer Graphics Forum*, volume 31, pages 519–528. Wiley Online Library, 2012.

[MCKM14] Matthias Müller, Nuttapong Chentanez, Tae-Yong Kim, and Miles Macklin. Strain Based Dynamics. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '14, pages 149–157, Aire-la-Ville, Switzerland, Switzerland, 2014. Eurographics Association.

[MFN16] Hammad Mazhar, Mihai Frâncu, and Dan Negruț. Simulating Large Scale Coupled Granular Material Simulations using Position Based Dynamics. In *The 4th Joint International Conference on Multibody System Dynamics*, 2016.

[MHHR07] Matthias Müller, Bruno Heidelberger, Marcus Hennix, and John Ratcliff. Position Based Dynamics. *J. Vis. Comun. Image Represent.*, 18(2):109–118, 2007.

[MHNT15] Hammad Mazhar, Toby Heyn, Dan Negruț, and Alessandro Tasora. Using Nesterov's Method to Accelerate Multibody Dynamics with Friction and Contact. *ACM Transactions on Graphics (TOG)*, 34(3):32, 2015.

[Mir96] Brian Vincent Mirtich. *Impulse-Based Dynamic Simulation of Rigid Body Systems*. PhD thesis, University of California at Berkeley, 1996.

[MM13] Miles Macklin and Matthias Müller. Position Based Fluids. *ACM Trans. Graph.*, 32(4):104:1–104:12, July 2013.

[MMCK14] Miles Macklin, Matthias Müller, Nuttapong Chentanez, and Tae-Yong Kim. Unified Particle Physics for Real-Time Applications. *ACM Transactions on Graphics (TOG)*, 33(4):104, 2014.

[Mor88] Jean J. Moreau. Unilateral Contact and Dry Friction in Finite Freedom Dynamics. In *Nonsmooth Mechanics and Applications*, pages 1–82. Springer, 1988.

[Mor05] Adam Moravánszky. NovodeX Demo Exercise. 2005.

[MSJT08] Matthias Müller, Jos Stam, Doug James, and Nils Thürey. Real Time Physics: Class Notes. In *ACM SIGGRAPH 2008 Classes*, pages 88:1–88:90, 2008.

[MSW14] Dominik L. Michels, Gerrit A. Sobottka, and Andreas G. Weber. Exponential Integrators for Stiff Elastodynamic Problems. *ACM Trans. Graph.*, 33(1):7:1–7:20, February 2014.

[MT10] Nadia Magnenat-Thalmann. *Modeling and Simulating Bodies and Garments*. Springer Science & Business Media, 2010.

[MTGG11] Sebastian Martin, Bernhard Thomaszewski, Eitan Grinspun, and Markus Gross. Example-Based Elastic Materials. *ACM Transactions on Graphics (TOG)*, 30(4):72, 2011.

[MTKV+11] Nadia Magnenat-Thalmann, Bart Kevelham, Pascal Volino, Mustafa Kasap, and Etienne Lyard. 3d Web-Based Virtual Try On of Physically Simulated Clothes. *Computer-Aided Design and Applications*, 8(2):163–174, 2011.

[Mül08] Matthias Müller. Hierarchical Position Based Dynamics. 2008.

[MW01] Jerrold E. Marsden and Matthew West. Discrete Mechanics and Variational Integrators. *Acta Numerica 2001*, 10:357–514, 2001.

[NGL10] Rahul Narain, Abhinav Golas, and Ming C. Lin. Free-flowing Granular Materials with Two-way Solid Coupling. In *ACM SIGGRAPH Asia 2010 Papers*, SIGGRAPH ASIA '10, pages 173:1–173:10, New York, NY, USA, 2010. ACM.

[NSO12] Rahul Narain, Armin Samii, and James F. O'Brien. Adaptive Anisotropic Remeshing for Cloth Simulation. *ACM Transactions on Graphics (TOG)*, 31(6):152, 2012.

[OAW06] Seungwoo Oh, Junghyun Ahn, and Kwangyun Wohn. Low Damped Cloth Simulation. *Vis. Comput.*, 22(2):70–79, 2006.

[OR70] James M. Ortega and Werner C. Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables*, volume 30. Siam, 1970.

[OR07] Ricardo Ortiz-Rosado. *Newton-AMG Algorithm for Solving Complementarity Problems Arising in Rigid Body Dynamics with Frictional Impacts*. ProQuest, 2007.

[Pai02] Dinesh K. Pai. Strands: Interactive Simulation of Thin Solids Using Cosserat Models. In *Computer Graphics Forum*, volume 21, pages 347–352. Wiley Online Library, 2002.

[PKMO02] A. Pandolfi, Couro Kane, Jerrold E. Marsden, and Michael Ortiz. Time-Discretized Variational Formulation of Non-Smooth Frictional Contact. *International Journal for Numerical Methods in Engineering*, 53(8):1801–1829, 2002.

[PNE10] Morten Poulsen, Sarah Niebe, and Kenny Erleben. Heuristic Convergence Rate Improvements of the Projected Gauss–Seidel Method for Frictional Contact Problems. pages 135–142. Václav Skala - UNION Agency, 2010.

[PO09] Eric G. Parker and James F. O'Brien. Real-Time Deformation and Fracture in a Game Environment. In *Proceedings of*

the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pages 165–175. ACM, 2009.

[PPR11] Tobias Preclik, Constantin Popa, and Ulrich Rüde. Regularizing a Time-Stepping Method for Rigid Multibody Dynamics. *Proceedings of Multibody Dynamics*, 2011, 2011.

[Pre08] Tobias Preclik. Iterative Rigid Multibody Dynamics, 2008.

[Pro96] Xavier Provot. Deformation Constraints in a Mass-Spring Model to Describe Rigid Cloth Behavior. In *In Graphics Interface*, pages 147–154, 1996.

[PS08] Jong-Shi Pang and David E Stewart. Differential Variational Inequalities. *Mathematical Programming*, 113(2):345–424, 2008.

[PT96] Jong-Shi Pang and Jeffrey C. Trinkle. Complementarity Formulations and Existence of Solutions of Dynamic Multi-Rigid-Body Contact Problems with Coulomb Friction. *Mathematical programming*, 73(2):199–226, 1996.

[PTVF07] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. Numerical Recipes in C. *Cambridge University Press*, 1:3, 2007.

[RA05] Mathieu Renouf and Pierre Alart. Conjugate Gradient Type Algorithms for Frictional Multi-Contact Problems: Applications to Granular Materials. *Computer Methods in Applied Mechanics and Engineering*, 194(18):2019–2041, 2005.

[RCB77] Jean-Paul Ryckaert, Giovanni Ciccotti, and Herman J. C. Berendsen. Numerical Integration of the Cartesian Equations of Motion of a System with Constraints: Molecular Dynamics of n-Alkanes. *J. Comput. Phys*, pages 327–341, 1977.

[RKC02] Stéphane Redon, Abderrahmane Kheddar, and Sabine Coquillart. Gauss' Least Constraints Principle and Rigid Body

Simulations. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, volume 1, pages 517–522. IEEE, 2002.

[Saa03] Yusef Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, 2nd edition, 2003.

[SB12] Eftychios Sifakis and Jernej Barbic. FEM Simulation of 3D Deformable Solids: A Practitioner's Guide to Theory, Discretization and Model Reduction. In *ACM SIGGRAPH 2012 Courses*, SIGGRAPH '12, pages 20:1–20:50, New York, NY, USA, 2012. ACM.

[SD06] Ari Stern and Mathieu Desbrun. Discrete Geometric Mechanics for Variational Time Integrators. In *ACM SIGGRAPH 2006 Courses*, SIGGRAPH '06, pages 75–80, New York, NY, USA, 2006. ACM.

[Sha09] Ahmed A. Shabana. *Computational Dynamics*. John Wiley & Sons, 2009.

[She94] Jonathan R. Shewchuk. An Introduction to the Conjugate Gradient Method Without the Agonizing Pain. Technical report, Pittsburgh, PA, USA, 1994.

[SHNE10a] Morten Silcowitz-Hansen, Sarah Niebe, and Kenny Erleben. A Nonsmooth jugate Gradient Method for Interactive Contact Force Problems. *The Visual Computer*, 26(6-8):893–901, 2010.

[SHNE10b] Morten Silcowitz-Hansen, Sarah Niebe, and Kenny Erleben. Projected Gauss-Seidel Subspace Minimization Method for Interactive Rigid Body Dynamics - Improving Animation Quality Using a Projected Gauss-Seidel Subspace Minimization Method. In Paul Richard, José Braz, and Adrian Hilton, editors, *GRAPP*, pages 38–45. INSTICC Press, 2010.

[Sho85] Ken Shoemake. Quaternions. 1985.

[SKV+12] Breannan Smith, Danny M. Kaufman, Etienne Vouga, Rasmus Tamstorf, and Eitan Grinspun. Reflections on Simultaneous Impact. *ACM Transactions on Graphics (TOG)*, 31(4):106, 2012.

[SLM06] Martin Servin, Claude Lacoursière, and Niklas Melin. Interactive Simulation of Elastic Deformable Materials. In *SIGRAD 2006 Conference Proceedings*, pages 22–32, 2006.

[Smi05] Russell Smith. Constraints in Rigid Body Dynamics. *Game Programming Gems*, 4:241–251, 2005.

[Smi06] Russell Smith. Open Dynamics Engine v0.5 User Guide. 2006.

[SS98] Jörg Sauer and Elmar Schömer. A Constraint-Based Approach to Rigid Body Dynamics for Virtual Reality Applications. In *Proceedings of the ACM symposium on Virtual Reality Software and Technology*, pages 153–162. ACM, 1998.

[SSB13] Funshing Sin, D. Schroeder, and Jernej Barbic. Vega: Non-Linear FEM Deformable Object Simulator. *Comput. Graph. Forum*, 32(1):36–48, 2013.

[SSC+13] Alexey Stomakhin, Craig Schroeder, Lawrence Chai, Joseph Teran, and Andrew Selle. A Material Point Method for Snow Simulation. *ACM Transactions on Graphics (TOG)*, 32(4):102, 2013.

[ST96] David Stewart and Jeffrey C. Trinkle. An Implicit Time-Stepping Scheme for Rigid Body Dynamics with Coulomb Friction. *International Journal for Numerical Methods in Engineering*, 39:2673–2691, 1996.

[Sta09] Jos Stam. Nucleus: Towards a Unified Dynamics Solver for Computer Graphics. In *Computer-Aided Design and Computer Graphics*, pages 1–11, 2009.

[Ste00] David E Stewart. Rigid-Body Dynamics with Friction and Impact. *SIAM review*, 42(1):3–39, 2000.

[Str07] Gilbert Strang. *Computational Science and Engineering*. Wellesley-Cambridge Press, 2007.

[Stu09] Christian Studer. *Numerics of Unilateral Contacts and Friction: Modeling and Numerical Time Integration in Non-Smooth Dynamics*, volume 47. Springer Science & Business Media, 2009.

[TA10] Alessandro Tasora and Mihai Anițescu. A Convex Complementarity Approach for Simulating Large Granular Flows. *J. Comput. Nonlinear Dynam.*, 5, 2010.

[TA11] Alessandro Tasora and Mihai Anițescu. A Matrix-Free Cone Complementarity Approach for Solving Large-Scale, Nonsmooth, Rigid Body Dynamics. *Computer Methods in Applied Mechanics and Engineering*, 200(5):439–453, 2011.

[TANN13] Alessandro Tasora, Mihai Anițescu, Stefano Negrini, and Dan Negruț. A Compliant Visco-Plastic Particle Contact Model Based on Differential Variational Inequalities. *International Journal of Non-Linear Mechanics*, 53:2–12, 2013.

[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.

[TKH+05] Matthias Teschner, Stefan Kimmerle, Bruno Heidelberger, Gabriel Zachmann, Laks Raghupathi, Arnulph Fuhrmann, M-P Cani, François Faure, Nadia Magnenat-Thalmann, Wolfgang Strasser, et al. Collision Detection for Deformable Objects. In *Computer graphics forum*, volume 24, pages 61–81. Wiley Online Library, 2005.

[TNGF15] Maxime Tournier, Matthieu Nesme, Benjamin Gilles, and François Faure. Stable Constrained Dynamics. *ACM Trans. Graph.*, 34(4):132:1–132:10, July 2015.

[Ton12] Richard Tonge. Solving Rigid Body Contacts. *Game Developer Conference*, 2012.

[TPS09] Bernhard Thomaszewski, Simon Pabst, and Wolfgang Straßer. Continuum-based strain limiting. *Comput. Graph. Forum*, 28(2):569–576, 2009.

[TSIHK06] Masayuki Tanaka, Mikio Sakai, Ishikawajima-Harima, and Seiichi Koshizuka. Rigid Body Simulation Using a Particle Method. In *ACM SIGGRAPH 2006 Research Posters*, SIGGRAPH '06, New York, NY, USA, 2006. ACM.

[TWS07] Bernhard Thomaszewski, Markus Wacker, and Wolfgang Straßer. Advanced Topics in Virtual Garment Simulation – Part 1, 2007.

[Ver67] Loup Verlet. Computer "Experiments" on Classical Fluids. I. Thermodynamical Properties of Lennard-Jones Molecules. *Physical review*, 159(1):98, 1967.

[VMT00] Pascal Volino and Nadia Magnenat-Thalmann. Virtual Clothing: Theory and Practice, 2000.

[VMTF09] Pascal Volino, Nadia Magnenat-Thalmann, and Francois Faure. A Simple Approach to Nonlinear Tensile Stiffness for Accurate Cloth Simulation. *ACM Trans. Graph.*, 28(4):105:1–105:16, 2009.

[WCF07] Ryan White, Keenan Crane, and David A. Forsyth. Capturing and Animating Occluded Cloth. In *ACM Transactions on Graphics (TOG)*, volume 26, page 34. ACM, 2007.

[WH91] Gerhard Wanner and Ernst Hairer. *Solving Ordinary Differential Equations II*, volume 1. Springer-Verlag, Berlin, 1991.

[Wit97] Andrew Witkin. Physically Based Modeling: Principles and Practice - Constrained Dynamics. *Computer graphics*, pages 11–21, 1997.

[WKMO99] Matthew West, Couro Kane, Jerold E. Marsden, and Michael Ortiz. Variational Integrators, the Newmark Scheme, and Dissipative Systems. In *International Conference on Differential Equations*, volume 1, page 2. World Scientific, 1999.

[WL06] Peter Wriggers and Tod A. Laursen. *Computational Contact Mechanics*, volume 30167. Springer, 2006.

[WN99] Stephen J Wright and Jorge Nocedal. *Numerical Optimization*. Springer New York, 1999.

[ZB05] Yongning Zhu and Robert Bridson. Animating Sand As a Fluid. In *ACM SIGGRAPH 2005 Papers*, SIGGRAPH '05, pages 965–972, New York, NY, USA, 2005. ACM.

[ZTZT77] Olgierd Cecil Zienkiewicz, Robert Leroy Taylor, Olgierd Cecil Zienkiewicz, and Robert Lee Taylor. *The Finite Element Method*, volume 3. McGraw-hill London, 1977.