Workshop on Virtual Reality Interaction and Physical Simulation VRIPHYS (2018)
F. Jaillet, G. Zachmann, K. Erleben, and S. Andrews (Editors)

Comparison of Mixed Linear Complementarity Problem Solvers for Multibody Simulations with Contact

Andreas Enzenhöfer[1,3], Sheldon Andrews[2], Marek Teichmann[3], and József Kövecses[1]

[1]Department of Mechanical Engineering, Centre for Intelligent Machines, McGill University, Canada
[2]Department of Software and IT Engineering, École de technologie supérieure, Canada
[3]CM Labs Simulations, Inc., Canada

Abstract

The trade-off between accuracy and computational performance is one of the central conflicts in real-time multibody simulations, much of which can be attributed to the method used to solve the constrained multibody equations. This paper examines four mixed linear complementarity problem (MLCP) algorithms when they are applied to physical problems involving frictional contact. We consider several different, and challenging, test cases such as grasping, stability of static models, closed loops, and long chains of bodies. The solver parameters are tuned for these simulations and the results are evaluated in terms of numerical accuracy and computational performance. The objective of this paper is to determine the accuracy properties of each solver, find the appropriate method for a defined task, and thus draw conclusions regarding the applicability of each method.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Physically Based Modeling—

1. Introduction

Physical simulations are an integral part of many interactive computer graphics applications. Virtual environments can often be described as multibody systems that are composed of rigid bodies. These rigid bodies model the physical components whose deformation is negligible compared to their overall displacement, and interactions between these components are represented by constraints and forces. A typical system includes both bilateral and unilateral constraints as well as friction at the contact interfaces. Bilateral constraints usually model joints, articulations and actuators, while unilateral constraints typically represent direct contact interactions between rigid bodies.

The work in this paper focuses on simulations involving both types of constraints, specifically where unilateral contact and friction between articulated rigid bodies is formulated as a *mixed linear complementarity problem* (MLCP) [DF95]. These systems present interesting numerical challenges due to non-smoothness introduced in the formulation by contacts, particularly with friction. The increasing complexity of physics-based virtual environments demands numerical solution methods that are accurate, robust, and efficient for a wide range of parameter settings such as mass, stiffness, and friction. Our objective is therefore to evaluate algorithms for solving constrained multibody problems formulated as MLCPs.

There are many off-the-shelf physics engines available for doing real-time simulation of articulated rigid bodies. However, the motion produced by these packages is dependent on the simulation settings. For example, some physics engines do poorly when the simulation involves large mass ratios or stiff joints. Such drawbacks are related to the underlying algorithm used to solve the constrained multibody problem. However, it can be difficult to choose an appropriate solver method for a specific physical problem.

Our motivation is not to help users pick an off-the-shelf simulation package, but rather to give an indication on what type of solver is most suitable for a particular type of task. Specifically, we consider four methods for solving constrained multibody problems. These are summarized in Section 3.2 and categorized into three main groups: direct (pivoting), indirect (iterative), and hybrid (combining the previous two approaches). We apply all methods to the same MLCP formulation using a box friction model (Section 3.1), and each algorithm is evaluated for its computational performance and numerical accuracy when solving challenging physical problems involving grasping, multi-contact interaction, stacking, and long kinematic chains such as cables.

2. Related Work

MLCPs are a generalization of the *linear complementarity problem* (LCP) [CPS92, MY88] where the variables can be subject to upper and lower bounds. Many authors review solution algorithms for the linear and nonlinear complementarity problem (CP) and compare the algorithm steps in theory [CPS92, Júd94, BDF97, Erl13, NE15, BET14, Lac07]. However, algorithm comparison in theory is often not sufficient to determine the resultant simulation accuracy and performance for particular problems. We discuss some examples of previous work comparing different open source and commercial

DOI: 10.2312/vriphys.20181063

dynamics simulation platforms such as Vortex Studio [CM 17b], ODE [Smi17], Bullet [Cou17], PhysX [NVI17], Havok [Hav17], and MuJoCo [Tod17]. Giovanni and Yin [GY11] evaluate and compare the performance of locomotion control on different simulation platforms via simulation results. Ivaldi et al. [IPPN14] conducted an online survey about simulation platforms in robotics. The study does not consider any simulation results but focuses on user feedback. Erez et al. [ETT15] compare the simulation results and determine the speed-accuracy trade-off for multiple physics engines starting with a small time step size and successively increasing the step size which leads to faster but less accurate simulations. The accuracy of the engine is measured by its level of self-consistency which is defined to be high if the deviation of the system configuration remains small for an increasing time step size. The comparison of different physics engines is very useful if one wants to choose an existing package. However, the dynamic formulations, i.e. the model of the constrained multibody system, used by the investigated platforms are likely to be substantially different from each other. MLCP solvers can only be compared in a meaningful way if the same dynamic formulation is used.

A benchmarking framework was developed specifically for rigid body dynamics with contact formulated as CPs and reported in [WLN*13, LLWT13, LT15, LWTL14]. The hierarchical data format (HDF5) is used to exchange data between multiple simulation platforms and load it into an analysis tool developed for the benchmarking project. Multiple direct and iterative solution algorithms, such as pivoting, matrix-splitting, and nonsmooth Newton methods, are tested on a variety of small and large-scale benchmark problems. The solver results are compared in terms of performance and accuracy. These comparisons eliminate the problem of different underlying models, however, there may still be differences in the dynamic formulations since the benchmarking framework allows the usage of linear and nonlinear CP formulations [LWTL14, LLWT13]. Furthermore, Lacoursière et al. [LLWT13] do not clarify if the presented simulation results are generated with the same CP formulation. The problem data is collected for each simulation time step [LWTL14], and the CPs are constructed based on the stored data so that every solution method receives the exact same CP to solve. This is crucial for a meaningful comparison since each solver can obtain a different solution due to numerical errors. For a continuous simulation with two different solvers (i.e. the solver result of the previous time step is used to formulate the CP for the next time step), the trajectories of the system configuration could start to deviate eventually. Then, we would not solve the same problem anymore. It can be seen as problematic that the number of iterations is used to quantify the solver performance since, for example, an iteration for an indirect method is usually much less expensive than an iteration for a direct method. This makes it difficult to compare performance results for these two types of methods. Furthermore, previous comparisons use MATLAB as a programming language, which introduces additional overhead thus making it difficult to obtain realistic timing information in the context of interactive computer graphics applications. Instead, we prefer to compare the computational time for solving a CP for each solution method implemented in the same software tool using the C++ programming language.

Drumwright and Shell [DS11] evaluate methods for modeling contact in multiple benchmark problems. The study quantifies the accuracy, performance, robustness, and speed of the methods which differ in dynamic formulation and solver. Drumwright and Shell [DS12] also perform an extensive analysis of LCP solver performance on randomly generated rigid body contact problems. Here, the LCPs are generated based on given properties for the mass matrix and constraint Jacobian so that each algorithm solves exactly the same LCP. However, these LCPs were not collected from the results of a continuous rigid body simulation. Therefore, we cannot relate an LCP with a specific scenario and system configuration. Neither of the two studies [DS11, DS12] includes iterative solvers based on matrix splitting schemes.

The solver comparison presented in this paper is inspired by multiple approaches [DS11, DS12, WLN*13, LLWT13, LT15, LWTL14] and follows the three principles:

- All solvers have to be applied to exactly the same MLCP.
- The MLCPs are obtained from the simulation of benchmark examples of physical problems.
- The solver performance is measured by the computational time needed to solve the MLCP.

3. Preliminaries

3.1. Box Friction Model

A well-known dynamic formulation for rigid body problems with unilateral contact and friction was introduced by Anitescu and Potra [AP97] as well as Stewart and Trinkle [ST96]. This approach uses a polygonal approximation of the friction cone and leads to a time stepping scheme at the impulse-velocity level. Hence, the nonlinear dynamic formulation is transformed into an MLCP which is proven to always have a solution if Lemke's algorithm is used [AP97]. However, this comes at the cost of discretizing the contact plane at each contact point with at least four tangent vectors. The box friction model [Lac06] requires only one normal and two tangent directions per contact point, thus three constraints. This decreases the number of constraints, and therefore the size of the MLCP, considerably in case of many contact points.

In the dynamic formulation used for the solver comparison in this paper, the constraints are regularized leading to the diagonal regularization matrices $\mathbf{C}$ and $\mathbf{D}$ dependent on the simulation time step size h as well as the user-specified stiffness and damping parameters k_i, b_i for each constraint i [Lac06]. The dynamic formulation for a rigid body system with regularized bilateral and unilateral constraint as well as friction can be written as

$$\begin{bmatrix} \mathbf{M} & -\mathbf{J}^{\mathrm{T}} \\ \mathbf{J} & \mathbf{C} \end{bmatrix} \begin{bmatrix} \mathbf{v}^{+} \\ h\boldsymbol{\lambda}^{+} \end{bmatrix} + \begin{bmatrix} -\mathbf{p} - h\mathbf{f}_a \\ \mathbf{D}\boldsymbol{\phi} \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{w} \end{bmatrix} \tag{1}$$

with mass matrix $\mathbf{M}$, momentum $\mathbf{p} = \mathbf{M}\mathbf{v}$, constraint Jacobian $\mathbf{J}$, constraint forces $\boldsymbol{\lambda}$, generalized velocities $\mathbf{v}$, applied forces $\mathbf{f}_a$, constraint violations $\boldsymbol{\phi}$, and constraint space velocities $\mathbf{w}$. All variables carrying the superscript $^{+}$ are evaluated at the next time step, all others at the current time step. The Schur complement of block $\mathbf{M}$ can be formed to remove the velocities from the MLCP [Erl07] which reduces the MLCP size further. This results in an MLCP with box bounds

$$\mathbf{A}\mathbf{x}+\mathbf{b}=\mathbf{w}_{+}+\mathbf{w}_{-}=\mathbf{w}, \tag{2a}$$

$$\begin{cases} \mathbf{0} \leq \mathbf{w}_{+} \perp \mathbf{x}-\boldsymbol{l} \geq \mathbf{0}, \\ \mathbf{0} \leq \mathbf{w}_{-} \perp \mathbf{u}-\mathbf{x} \geq \mathbf{0}, \end{cases} \tag{2b}$$

where $\mathbf{A}=(\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^{\mathrm{T}}+\mathbf{C}) \in \mathbb{R}^{n\times n}$ is the MLCP lead matrix, $\mathbf{b}=\mathbf{J}\mathbf{M}^{-1}(\mathbf{p}+h\mathbf{f}_{a})+\mathbf{D}\boldsymbol{\phi} \in \mathbb{R}^{n}$ is given, $\mathbf{x}=h\boldsymbol{\lambda} \in \mathbb{R}^{n}$ are the constraint impulses, and $\mathbf{u}$ and $\boldsymbol{l}$ are upper and lower bounds on the constraint impulses, respectively. We subdivide $\mathbf{w}$ into nonnegative, complementary components, i.e. $\mathbf{0} \leq \mathbf{w}_{+} \perp \mathbf{w}_{-} \geq \mathbf{0}$.

3.2. Mixed Linear Complementarity Problem Solvers

We implement four algorithms for solving the MLCP arising from the box friction model: the *Block Principal Pivoting* (BPP) algorithm [JP94], *Projected Gauss-Seidel* (PGS) [Erl07], *Projected Gauss-Seidel with Subspace Minimization* (PGS-SM) [SNE10], and the *Spook Stepper* (SPOOK) [LL11]. We briefly summarize these methods and our minor modifications in the following section and refer the reader to the above papers for more detailed information.

3.2.1. Block Principal Pivoting

BPP is the only purely direct method amongst the ones presented above. It computes an accurate solution by systematically determining the index sets. These are two sets providing a label for each variable as basic or non-basic. Non-basic variables are assumed to be known, basic variables are unknown. The algorithm convergence rate is varying and is only guaranteed in exponential time when BPP switches to single pivoting [JP94], i.e. the method can find a solution to the MLCP after only a few or many pivoting steps depending on the problem. There is no guarantee for closeness of any temporary solution of an iteration to the final one. Each pivoting step has complexity $\mathcal{O}(n^3)$ using dense lead matrix factorizations and could be sped up to $\mathcal{O}(n^2)$ for band matrices with a small upper and lower envelops. In each pivoting step, the values of the set of basic variables is solved using a direct approach. Any variables outside the bounds are pivoted to the non-basic set. BPP terminates if there are no more changes in the index sets, i.e. none of the variables are out of bounds.

3.2.2. Projected Gauss-Seidel

PGS is an iterative algorithm that breaks up Eq. (2a) into n single equations using a matrix splitting technique and setting $\mathbf{w}$ to zero. The solution is approximated by performing fixed-point iterations and projecting the solution onto the bounds. An iteration is computationally far less expensive than a pivoting step: $\mathcal{O}(n)$ when exploiting sparsity of the lead matrix by performing floating point operations only on non-zero matrix elements. This holds if the number of non-zero matrix elements is in the same order as the matrix size n which is the case for large-scale multibody dynamics systems. However, convergence is reported to be linear at best [Erl05, Lac07]. Convergence is guaranteed for $\rho(\mathbf{A}) < 1$, where ρ is the spectral radius. In contrast to Erleben's PGS solver [Erl07], we do not update the friction bounds after solving for the normal row of a contact point so that the same MLCP is solved and the algorithm results are comparable. We check for convergence by computing the residual of each constraint. The algorithm stops if the maximum error of all variables drops below a user-defined tolerance threshold.

3.2.3. Hybrid Methods

The final two solvers, SPOOK and PGS-SM, are hybrid approaches combining direct and iterative methods. PGS-SM uses an iterative PGS phase to estimate the index sets, and then proceeds to a subspace minimization phase to solve the basic set. Our implementation uses a Cholesky factorization to solve the basic set, which may then be further reduced by pivoting variables that are out of bounds before returning to the PGS phase to re-estimate the index sets. This cycle continues until convergence is reached, i.e. for our implementation that there are no more changes in the index sets.

Similarly, SPOOK uses a splitting approach by applying a direct solver to compute the constraint forces of bilateral and normal contact constraints, together. Then, an iterative method is used to solve for contact normal constraints and friction forces together. In our implementation, the direct phase uses the previously described BPP solver, and the iterative phase consists of a blocked Gauss-Seidel method [BET14], i.e. we solve for the normal and friction forces together using the 3x3 diagonal blocks corresponding to each contact and iterate over all blocks. Direct and iterative phases alternate, and we terminate with a direct solve if there are no more changes in the index sets.

4. Test Setup

A major problem comparing the simulation results of different solvers is the divergence from the true solution in case of simulation error. If two solvers obtain two different results, the system configuration and motion are not the same in the beginning of the next time step nor is the resultant MLCP. It can happen that the MLCP for one of the solvers is much less complex, e.g. when contact points detach, which can lead to comparisons of limited meaning. This issue can be prevented by creating a reference MLCP for each time step and solving this reference MLCP using all solution algorithms to be compared. First, the reference MLCPs are created by solving all test cases in Section 5 using BPP without enforcing any time or iteration limit, called *reference solver*. The use of a convergent direct solver guarantees that the solution is always an accurate result of the MLCP and that there is no simulation error caused by the solver, i.e. no divergence from the true solution of the MLCP. Note that there may still be an error introduced by the friction model or the time discretization, however, we do not intend to measure nor analyze these types of error in this paper. Second, the reference MLCPs are solved by each of the four solvers in this comparison. This guarantees that we have the same starting point for each solver in the same time step. The accuracy of the solution obtained by the solver is then measured using the solver error described in the following paragraph, not the solution computed by the reference solver. We focus on two essential measures: numerical accuracy and computational speed. We often cannot achieve both at the same time. Hence, a substantial part of choosing a solver and its parameters for a particular type of problem is to quantify the trade-off between these two properties.

Type of constraint	Stiffness k_i	Damping b_i
Bilateral	$10^{10}\ \frac{N}{m}$	$10^{8}\ \frac{kg}{s}$
Unilateral	$10^{5}\ \frac{N}{m}$	$10^{4}\ \frac{kg}{s}$

Table 1: *Constraint stiffness and damping parameters for all test cases*

Problem size	Grasping	Brick wall	Winch	Closed loop
n	~ 400	~ 550	~ 400	~ 350

Table 2: *Average number of constraints, variations due to contact detachment*

Solver accuracy: The accuracy of the solver is inversely proportional to the *solver error*. We measure the solver error by computing the *natural residual* [Pan86,MS86]. We determine for each pair of components whether the constraint impulse x_i or the related constraint space velocity w_i violate the constraint conditions in Eq. (2b) and compute the natural residual for each row to form the residual component vector $\boldsymbol{\delta\psi}_{\text{res}} = [\delta\psi_{\text{res},1} \dots \delta\psi_{\text{res},n}]^{\mathrm{T}}$. Then, the ℓ_1-norm of $\boldsymbol{\delta\psi}_{\text{res}}$ can be used to define the error of the system as

$$\delta\psi_{\text{res},i} = \max\left\{\left|\min\left(x_i - l_i, w_{i,+}\right)\right|, \left|\min\left(u_i - x_i, w_{i,-}\right)\right|\right\}, \quad (3a)$$

$$\delta\psi_{\text{res}} = ||\boldsymbol{\delta\psi}_{\text{res}}||_1 = \sum_{i=1}^{n} |\delta\psi_{\text{res},i}|. \quad (3b)$$

In principle, any type of vector norm could be used to obtain the system error. We choose the ℓ_1-norm because it is simply the sum of all components given all components are positive.

Solver performance: The performance of the solver is inversely proportional to the computational time the solver needs to find a solution, here also called *solver time*. We measure the solver time for each method using a high-resolution timer for the CPU time, excluding collision detection. Furthermore, we solve the exact same MLCP multiple times and compute the average solver time of all executions.

5. Test Cases

In this section, we describe the examples used to evaluate each solver: grasping with a claw, simulating a winch, a closed-loop structure, and a brick wall. Table 1 shows the constraint relaxation parameters used for all test cases. The friction coefficient is set to $\mu = 1.0$ and gravity $g = 9.81$ N/kg, which acts in a direction perpendicular to the ground plane. The integration time step is chosen to be $h = 1/60$ s and 500 simulation steps are executed for each example. All problems require solving large-scale MLCPs. The approximate problem size n, which equals the number of constraints, is given in Table 2. Note that n fluctuates for different time steps due to newly arising or detaching contacts.

5.1. Log Grasping

This example, which is illustrated in Figure 1, contains a claw gripper attached to an overconstrained forwarder arm consisting of 12 revolute, 2 prismatic, and 2 cylindrical joints that connect 13 bodies. The claw grasps a log at its center of mass, and the gripper arm lifts the trunk and rotates it around the vertical direction. This scenario combines some of the challenging parts of grasping: First, a stable grasp around the log needs to be established. Second, the log is lifted without sliding through the claw. Third, a non-negligible friction force is acting between log and claw due to the arm rotation, and an incorrect solve of the friction forces may lead to dropping the log. Note that the vehicle, which the gripper arm is attached to, is not considered in the performance and accuracy measurements.

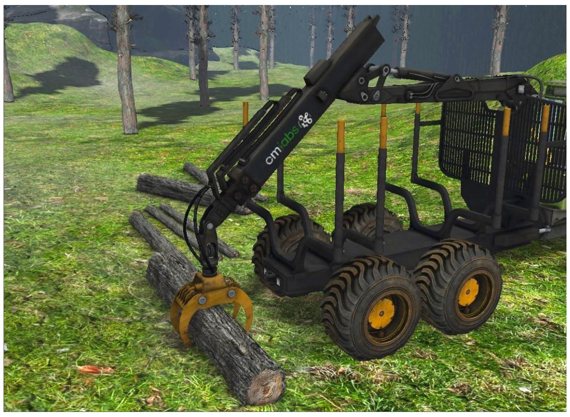


Figure 1: *Test case (a) - log grasping*

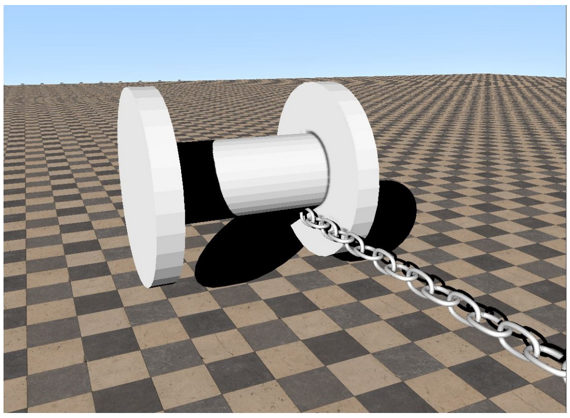


Figure 2: *Test case (b) - winch*

5.2. Winch

A chain, with links modeled using capsules and connected by spherical joints, is resting on the ground. One of its ends is attached to an articulated winch, which is located slightly above the ground. The winch rotates at a constant velocity of 160 revolutions per minute, in order to wind the chain of 50 segments (Figure 2). The chain is initially in fully horizontal position connected to the winch and falls to the ground when the simulation is started. Furthermore, the chain is dragged along the ground before reaching the winch. This system is of interest as it contains large constraint

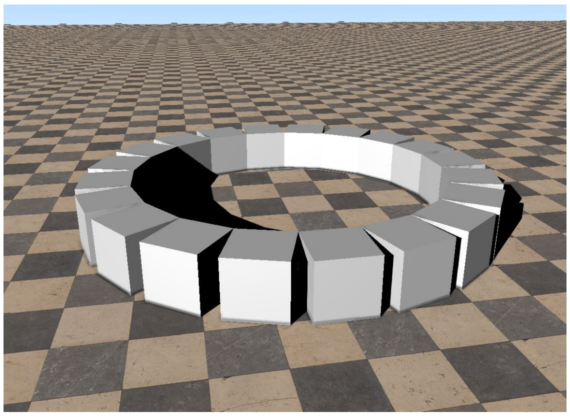


Figure 3: *Test case (c) - closed loop*

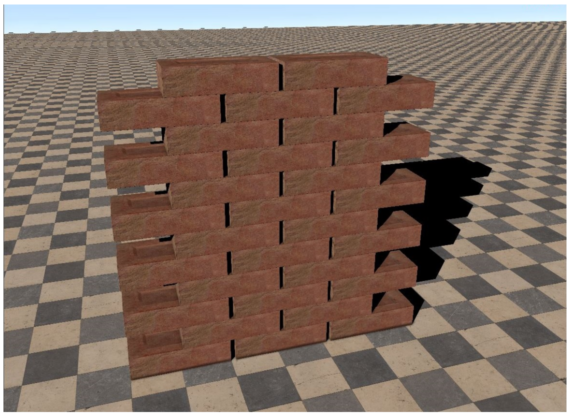


Figure 4: *Test case (d) - brick wall*

forces along the chain that would pull segments apart, which can lead to instability.

5.3. Closed Loop

Shown in Figure 3, 20 cubes are arranged in a circle on the ground and connected by revolute joints, which forms a closed loop of bodies. This example is highly redundant due to having multiple contacts between the ground and each cube and due to the revolute joints which are kinematically aligned. The loop is dropped on the ground from an initial height equal to the box dimensions. Additionally, we apply tangential forces of 100 N to each of the bodies in every step before the loop reaches the ground in order to make the loop spin fast around its symmetry axis perpendicular to the ground. Once the loop impacts with the ground, friction forces counteract the spinning motion and the loop comes to rest eventually.

5.4. Brick Wall

This example consists of a stack of 30 boxes laid out in a brick wall pattern (Figure 4). The brick wall is 12 bricks tall, and we alternate between rows of two or three bricks in width. Furthermore, there are small horizontal and vertical gaps between two bricks in the initial configuration so that there is no contact in the beginning of the simulation. This example is meant to investigate how each algorithm preserves stability for a static problem, and deals with simulating environments where contact rich interactions may occur.

6. Results

All MLCP solvers are implemented in C++ using the Vortex Dynamics engine for collision detection [CM 17b, CM 17a] . The simulations are executed using an Intel Core i7-6700HQ processor with 3.50 GHz and 6MB cache. A single threaded implementation in double precision is used for each algorithm, and the physical problem is solved without partitioning. Iterative solvers exploit the sparsity of the lead matrix by performing floating point operations only on non-zero matrix elements leading to a complexity of $\mathcal{O}(n)$ per iteration for large sparse systems. Direct solvers use dense matrix representations and a Cholesky factorization operating on the MLCP matrix leading to a complexity of $\mathcal{O}(n^3)$ per iteration.

6.1. Tuning for Convergence

We begin by tuning the iteration limit for each solution algorithm. To find the optimal values, we gradually increase the parameter until convergence is achieved, or the algorithm stagnates, i.e. no progress is being made to reduce the error. The solver error for each iteration is stored, and this information helps to determine the convergence of iterative solvers and to find an appropriate tolerance threshold. The histograms illustrating the number of iterations required to reach convergence, i.e. stagnation in the index sets, are shown in Figure 5. Time steps for which the solver is not able to find a solution in the given iteration limit are collected in the hatched red bars.

In most examples, the BPP solver terminates within a small number of steps, which is less than 10 pivoting steps for the majority of investigated time steps (shown for grasping in Figure 5 (a)). Since SPOOK uses a pivoting method to compute only bilateral constraint and contact normal forces, the system to be solved during the direct phase is considerably smaller. This avoids pivoting the upper and lower bounded friction rows, which seem to be the most problematic, and explains why the direct phase of SPOOK takes less than 5 pivoting steps in most cases (visualized for grasping in Figure 5 (d)). Likewise, the direct phase of PGS-SM tries to reduce the size of the basic set at each step, and never increases it [SNE10]. PGS-SM finishes reducing the size of the basic set after at most 5 direct iterations, as shown for grasping in Figure 5 (b). Coupling between the iterative and direct phase is necessary to add new variables to the basic set. Convergence is reached in 6 iterations or less for about 90% of all tested time steps illustrated in Figure 5 (c) for grasping.

For iterative methods, it is difficult to obtain an accurate solution within a reasonable number of iterations due to the linear convergence of the PGS algorithm. This can be observed by looking at Figure 6 and noting that PGS diverges for the grasping example, regardless of the maximum iteration count (Figure 6 (a)); it converges for the other test cases. However, it is not possible to reduce

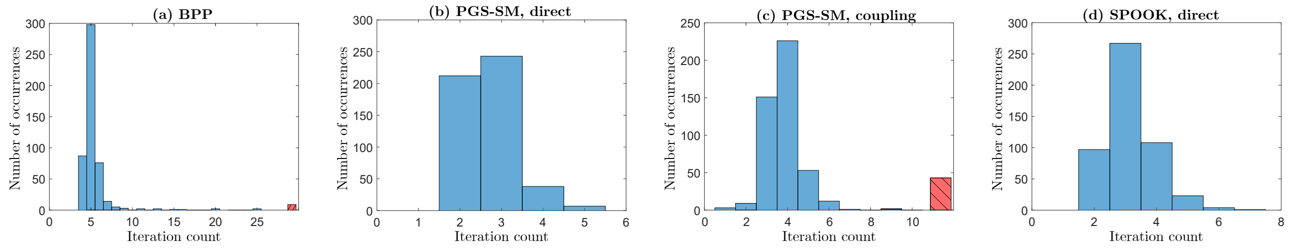
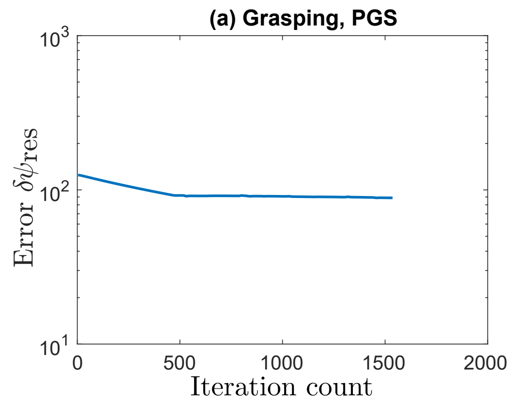


Figure 5: *Iteration count for grasping using BPP ((a): direct), PGS-SM ((b): direct, (c): coupling), SPOOK ((d): direct); number of occurrences of the corresponding iteration counts in 500 tested time steps; the hatched red bars illustrate cases for which no solution was found within the iteration limit*



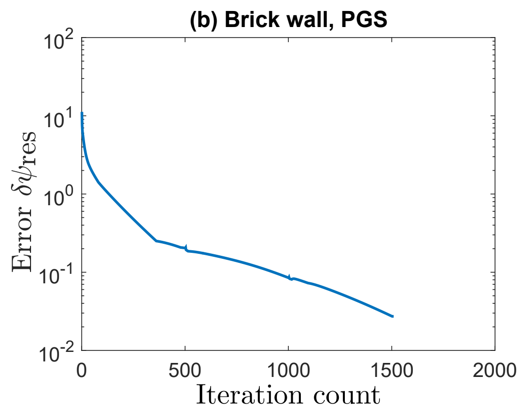


Figure 6: *Development of the solver error $\delta\psi_{res}$ in a single time step executed with PGS for (a) grasping (divergent solver) and (b) the brick wall (convergent solver)*

the solver error below 10^{-2} without significantly exceeding real time (Figure 6 (b)). In SPOOK, friction forces are only updated in the iterative phase. Thus, it has difficulties to obtain an accurate friction solution whereas the bilateral and unilateral constraint forces show the same precision as BPP and PGS-SM if they do not change substantially for variations in the friction forces.

6.2. Default Solver Parameters

Table 3 shows the default solver parameters concluded from the above analysis. We chose the log grasping example to determine the default parameters since it is most sensitive to simulation instabilities in case of large errors and was most difficult to tune. Two parameters are given for the iteration count limit: k_{dir} for direct and k_{iter} for iterative solvers. We choose different convergence criteria for direct and iterative algorithms independent of the measured solver error. Direct methods keep track of the index sets of all components. Changes in the solution can only occur if the index sets change. Thus, the algorithms terminate if the index sets stagnate. Iterative methods gradually converge to the solution by updating the constraint reaction forces in every iteration. The iterative solver stops if the maximum change per force component with respect to the previous iteration drops below the tolerance ε. Thus, Table 3 lists a tolerance value for all solvers but BPP. We choose comparably small value for ε to guarantee that the algorithm does not stop before a solution with a reasonably small error is obtained or the iteration limit is reached.

The solver accuracy and performance depend on multiple factors such as the dynamic formulation, friction model and regularization parameters. Hence, it is recommended tuning these parameters specifically for one's simulation environment. The tolerance threshold is decisive to determine the convergence of an iterative method or the iterative phase of a hybrid algorithm. If it is set too small, a solver may be prevented from terminating and keep iterating without significantly improving the solution in terms of the solver error. On the other hand, a large threshold leads to a relatively large solver error which increases the risk of divergence from reality in the next time steps.

The number of iterations is limited for each solver and algorithm phase in order to perform real-time or close to real-time simulations. Especially for direct solvers, this can lead to large errors in the solution if the solver does not terminate. The upper and lower friction bounds are updated when the solver reaches the iteration limit or finds the solution for the current bounds. After each update, the algorithm iteration count is reset, e.g. PGS performs a maximum of $k_{\text{cpl}} \cdot k_{\text{iter}} = 75$ overall iterations for $k_{\text{cpl}} = 3$ coupling iterations and $k_{\text{iter}} = 25$ solver iterations. The coupling iterations are necessary to alternate between the direct and iterative phases of PGS-SM and SPOOK as well as to update the friction bounds for BPP and PGS. All methods are warm-started after a bound update.

	BPP	PGS	PGS-SM	SPOOK
Coupling iterations	$k_{\text{cpl}} = 3$	$k_{\text{cpl}} = 3$	$k_{\text{cpl}} = 5$	$k_{\text{cpl}} = 5$
Iteration limits	$k_{\text{dir}} = 30$	$k_{\text{iter}} = 25$	$k_{\text{iter}} = 15$ $k_{\text{dir}} = 5$	$k_{\text{iter}} = 15$ $k_{\text{dir}} = 10$
Termination criteria	Change in index sets	Change in forces	Change in index sets & forces	Change in index sets & forces
Tolerance	-	$\varepsilon = 10^{-5}$	$\varepsilon = 10^{-5}$	$\varepsilon = 10^{-5}$

Table 3: *Solver parameters: number of coupling iterations, iteration limit for each solver phase, termination criteria, tolerance*

6.3. Test Case Results

The plots in Figure 7 show the simulation results for all test cases in Section 5. We choose a logarithmic scale for the measured average solver time on the x-axis and the energy error on the y-axis. Every data point represents one of the time steps. The simulation results are visualized in the supplementary video accompanying this paper. All solvers are displayed in parallel for every test case. In addition, the performance differences are outlined in a simulation of the winch example recorded at interactive frame rates where the CPU time for some solvers is greater than the frame rate so that the simulation slows down.

6.3.1. Log Grasping

The log grasping results in Figure 7 (a) show the largest errors for all solvers compared to the other test cases as well as comparably high solver times. The error for PGS is so high that the bilateral constraints are not satisfied anymore so that the gripper arm and the vehicle visually collapse as shown in the video. This behavior can be explained by the large mass ratios and stiff constraints of the system, typically a shortcoming of PGS. SPOOK cannot maintain a stable grasp so that the log starts sliding through the gripper and is nearly dropped due to large errors in the friction forces. BPP and PGS-SM are able to perform the grasping operation but do not achieve real time. Except for a few time steps when no convergence can be reached within the iteration limit, BPP reaches consistently low errors with solver times fluctuating between 10^{-2} s and 10^{0} s. PGS-SM shows less fluctuations in solver time but larger solver errors which cannot be visually perceived. A direct solve of the MLCP representing the entire system, such as in BPP and PGS-SM, is needed to obtain the required accuracy for this grasping task. Note that the measured solver time contains only the computational time for the gripper and gripper arm but not the vehicle.

6.3.2. Winch

For the winch examples in Figure 7 (b), we can observe multiple steps showing low computational times and solver error for all solvers. In the beginning of the simulation, the MLCP size is significantly smaller and there are no unilateral constraints or friction if the chain does not touch the ground. The performance drops between one or two orders of magnitude when the chain then reaches the ground. Solver errors are similar for PGS and SPOOK with better performance values for PGS. BPP and PGS-SM lead to similar error magnitudes while PGS-SM is faster on average. For BPP and PGS-SM, there are a few data points showing solver errors around 10^{0} which occurred in cases where the solvers did not converge for the given iteration limit. The interactively captured video clearly shows the solver performance in descending order: PGS, SPOOK, PGS-SM, BPP. We recommend using PGS for fast, less accurate solutions and PGS-SM for accurate solutions when more computational time is available.

6.3.3. Closed Loop

Similar to the winch, there are multiple time steps showing much lower solver times and errors than average for the closed loop in Figure 7 (c). These time steps occur before the loop reaches the ground. BPP has the lowest error in this test but also the worst performance for the time steps during which the loop is in contact with the ground. The error for PGS-SM increases several orders of magnitude but the method leads to simulations slightly faster then BPP. The video shows no visual differences in the motion of the loop. PGS and SPOOK obtain similar solver times but less error for SPOOK. The BPP method is recommended if the solution needs to be accurate and SPOOK if performance has priority.

6.3.4. Brick Wall

The brick wall is the only motionless example among the test cases. Therefore, solver times and errors should be very similar for all steps as the system configuration should not change much throughout the simulation. This appears to be true for most of the simulation, however, there are some time steps requiring less CPU time. This can be explained by monitoring the number of constraints per time step, i.e. the MLCP size. Initially, the bricks are not in contact so that they simply fall due to gravity and we do not need to solve an MLCP. When the first row of bricks reaches the ground, contact is detected leading to an MLCP and contact forces are applied to the bricks. Then, the number of contacts as well as the MLCP size increases when the next row of bricks impacts with the previous one leading to an increase in solver time. Eventually, all contacts are closed so that the solver time reaches its maximum. The video illustrates that the bricks are slightly misplaced for PGS-SM which also shows in the relatively large solver errors for some time steps (Figure 7 (d)). PGS and SPOOK both run in real time but SPOOK is able to keep the brick wall stable beyond 1000 time steps whereas the wall collapses eventually using PGS. We recommend using SPOOK for high performance and BPP for high accuracy.

6.4. Median Results

Table 4 contains the median of the solver error and solver time regarding all 500 time steps given for each test case and MLCP solver. We choose to compute the median error and time rather than the mean to find the "typically expected" solver error. The median leads to the disregard of of high and low peaks. BPP obtains similar computational solver time for all examples and solver errors below 10^{-10} except for grasping. The high median solver error of 10^{2} for PGS indicates the simulation failure for the grasping task. PGS obtains the lowest median error for the closed loop and leads to the lowest solver time for all examples but the brick wall. PGS-SM computes accurate results for all examples, the error is higher for grasping which does not show if we visually observe the simulation. SPOOK suffers from inaccurate friction forces which increases the overall error in all test cases.

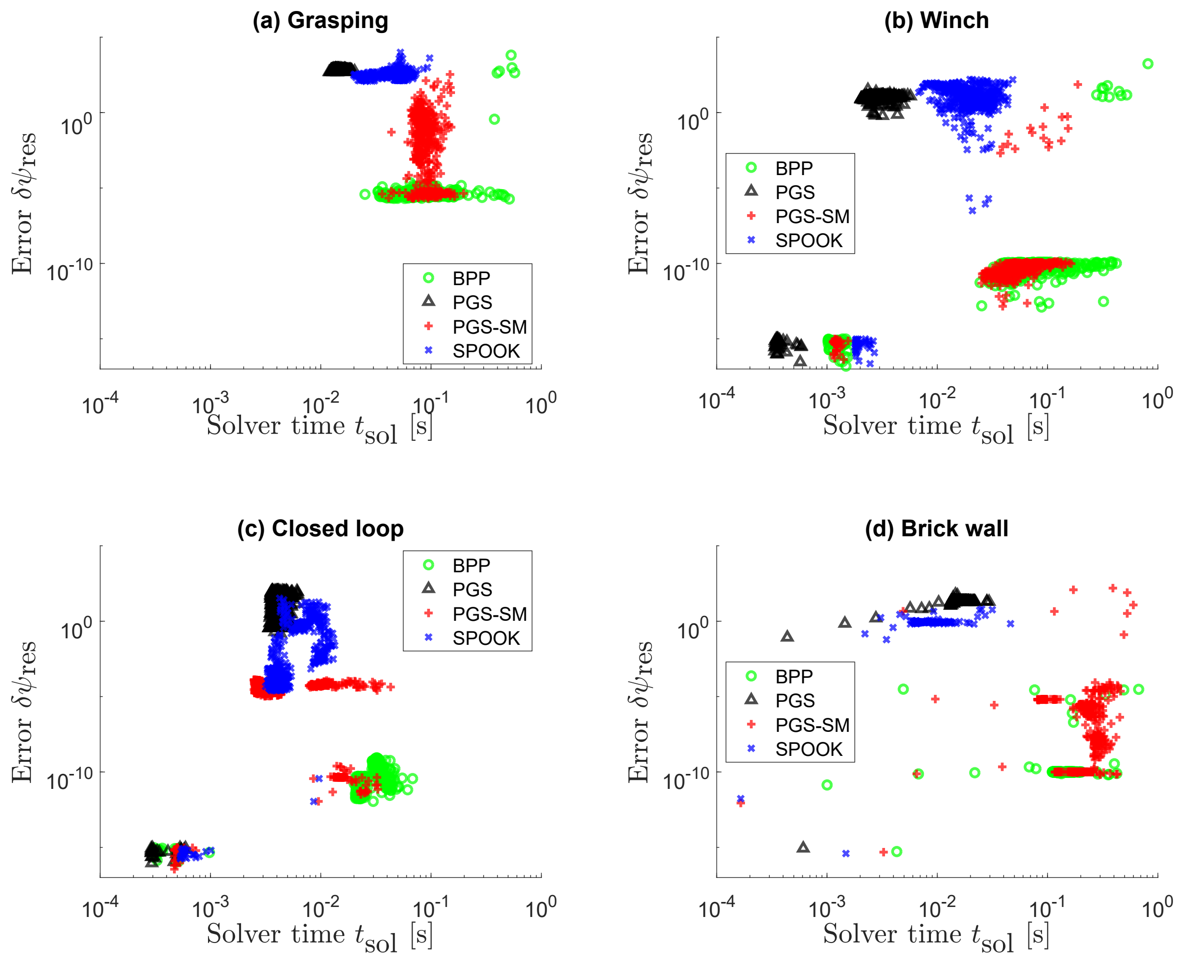


Figure 7: *Solver error defined by the natural residual $\delta\psi_{res}$ in Eq. (3); solver time t_{sol} defined by the CPU time required to solve the MLCP; 500 time steps for each scenario: (a) log grasping, (b) winch, (c) closed loop, (d) brick wall*

	Grasping		Winch		Closed loop		Brick wall	
Solver	$\delta\psi_{\text{res}}$	t_{sol} [s]	$\delta\psi_{\text{res}}$	t_{sol} [s]	$\delta\psi_{\text{res}}$	t_{sol} [s]	$\delta\psi_{\text{res}}$	t_{sol} [s]
BPP	10^{-6}	$8.46 \cdot 10^{-2}$	10^{-11}	$5.42 \cdot 10^{-2}$	10^{-11}	$2.38 \cdot 10^{-2}$	10^{-10}	$1.46 \cdot 10^{-1}$
PGS	10^{2}	$1.43 \cdot 10^{-2}$	10^{1}	$2.66 \cdot 10^{-3}$	10^{-1}	$3.88 \cdot 10^{-3}$	10^{1}	$1.45 \cdot 10^{-2}$
PGS-SM	10^{-2}	$8.68 \cdot 10^{-2}$	10^{-11}	$5.12 \cdot 10^{-2}$	10^{-5}	$8.52 \cdot 10^{-3}$	10^{-8}	$2.26 \cdot 10^{-1}$
SPOOK	10^{2}	$4.89 \cdot 10^{-2}$	10^{1}	$1.85 \cdot 10^{-2}$	10^{-3}	$4.52 \cdot 10^{-3}$	10^{-1}	$6.45 \cdot 10^{-3}$

Table 4: *Median of all data points in Figure 7 for the solver error $\delta\psi_{res}$ and the solver time t_{sol}*

6.5. Discussion

In our experiments, we observe some other noteworthy behavior.

Constraint ordering. We notice the common phenomenon for iterative methods that convergence and solution accuracy are highly dependent on the order which the constraint are solved in [AEKT17]. Therefore, computation of the contact forces for the bricks closer to the ground first leads to lower errors and better convergence of an iterative solver.

Insufficient friction. We note that the SPOOK solver often fails to produce adequate friction forces. In a further experiment, we model a cable as a serial chain of rigid bodies which is then wrapped around a rod modeled by a single rigid cylinder. Both ends of the cable are pulled to create tension while the rod is pulled out of the sling formed by the chain. We expect to need a large force to separate the rod from the cable due to the substantial friction forces along the rod axis. However, SPOOK does not produce sufficient friction forces unless we significantly increase the number of cou-

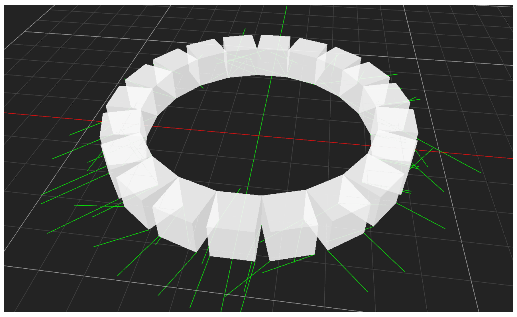


Figure 8: *A debug view of the closed loop simulation showing friction forces (green lines). After the mechanisms slides and comes to rest, the BPP and SPOOK solvers sometimes produce incorrect frictional forces.*

pling iterations between the direct and iterative phase since many iterations are required to capture the correct coupling between the bilateral cable constraints, contact normals, and friction.

Stationary point. We observe that the BPP and SPOOK solvers sometimes converge to a stationary point. For instance, in the case of the highly over-constrained example of the closed loop, these solvers would produce an MLCP solution with non-zero friction forces even when the loop is at rest and there are no external forces other than gravity. The resulting motion is correct, since the friction forces produce zero resultant force and torque, but physically this behavior is obviously incorrect. We find that this phenomenon (shown in Figure 8) can be mitigated in the SPOOK solver by introducing a regularization term for the frictional constraints early during the iterative phase. The term is gradually reduced at each iteration until it eventually reaches zero. This biases the solver to minimize frictional forces while still solving the MLCP.

Lowest error solution. Direct methods compute an accurate solution of the MLCP if the algorithm terminates. However, if the algorithm does not terminate, e.g. due to an iteration limit common in real-time simulations, the result obtained in the last iteration is in general not the "best" solution, i.e. the closest to a true solution of the MLCP. Thus, we recommend storing the best solution over all iterations and use it to compute the system motion and configuration at the next time step. This can significantly reduce the risk of simulation instabilities. In this paper, we use the natural residual in Eq. (3) to determine the solver error which represents the closeness of a computed solution to a true solution of the MLCP. However, this definition of the solver error can be problematic, for example, if there are high mass ratios in the system. In this case, two constraint forces of similar magnitude have substantially different effects on a body of high or low mass. Then, a relatively small constraint force error can lead to large error in the acceleration of the light body whereas a relatively large constraint force error can cause a small error in the acceleration of the heavy body. Choosing the solution with the least force error will thus not necessarily keep the system stable. Furthermore, the natural residual is unit inconsistent if some constraint errors lie in the constraint space acceleration, others in the constraint forces, so that there is no physical meaning in the sum of these errors. Both issues can be prevented by scaling the solver error with the effective mass perceived by the constraint which is researched in ongoing work.

7. CONCLUSIONS

BPP and PGS-SM are accurate solution methods leading to low solver errors because both methods terminate with a direct solve for the MLCP including bilateral and unilateral constraints as well as friction. If accuracy has priority over performance, BPP and PGS-SM are the preferred methods to obtain accurate simulation results. However, direct solvers may require a substantial number of iterations before computing an accurate solution and should therefore only be used if convergence is reached for the vast majority of time steps. PGS solves the MLCP iteratively which requires comparably little computational time. Thus, the method should be used for simulations where accuracy is secondary but little computational time is available for the solver. In contrast to direct solvers, the result of a PGS iteration is always guaranteed to be closer to the solution than the result of the previous iteration given the algorithm converges. SPOOK solves for unilateral constraints and friction iteratively while applying a pivoting method to the bilateral and unilateral constraints to prevent interpenetration. This delivers accurate results for the bilateral and unilateral constraints if these constraint forces do not change significantly in case of errors in the friction forces. SPOOK is recommended in cases where the friction forces are not decisive for the system configuration. If small friction changes have a significant impact on bilateral and unilateral constraint forces, SPOOK should not be used.

7.1. Future Work

In future work, we intend to optimize our algorithm implementations to achieve faster computational speed. Sparse matrix representation and variable bandwidth factorizations for the MLCP lead matrix in direct methods can decrease algorithm complexity significantly and thus can lead to performance improvements. Moreover, instead of recomputing the matrix factorization at each algorithm step, as in the BPP and PGS-SM computation of the active set, existing factorizations can be updated or downdated efficiently if only a few index sets have been pivoted.

Another interesting avenue for future work is to explore the dependency between the MLCP solver and the success rate of controllers in character animation or robotics simulation for locomotion and grasping tasks. Generally, iterative solvers may have difficulties to compute accurate constraint forces for these tasks so that the system motion may seem visually plausible but is not physically correct. A direct solver used for critical subsystems in a simulation can help to increase the accuracy and thus lead to physical correctness of the simulation. We also plan to expand the repertoire of solvers used in our experiments.

Acknowledgements

This work was supported by the National Science and Engineering Research Council of Canada (NSERC) and CM Labs Simulations Inc. The authors would like to thank the Vortex Dynamics team around Daniel Holz at CM Labs Simulations for their help with the solver implementations.

References

[AEKT17] ANDREWS S., ERLEBEN K., KRY P. G., TEICHMANN M.: Constraint reordering for iterative multi-body simulation with contact. In *ECCOMAS Thematic Conference on Multibody Dynamic* (Prague, Czech Republic, June 18–22, 2017). 8

[AP97] ANITESCU M., POTRA F. A.: Formulating dynamic multi-rigid-body contact problems with friction as solvable linear complementarity problems. *Nonlinear Dynamics 14*, 3 (1997), 231–247. 2

[BDF97] BILLUPS S. C., DIRKSE S. P., FERRIS M. C.: A comparison of large scale mixed complementarity problem solvers. *Computational Optimization and Applications 7*, 1 (1997), 3–25. 1

[BET14] BENDER J., ERLEBEN K., TRINKLE J.: Interactive simulation of rigid body dynamics in computer graphics. *Computer Graphics Forum 33*, 1 (2014), 246–270. 1, 3

[CM 17a] CM LABS SIMULATIONS: Theory guide: Vortex software's multibody dynamics engine. https://www.cm-labs.com/vortexstudiodocumentation/Vortex_User_Documentation/Content/Concepts/Vortex_Dynamics_Theory_final.pdf, 2017. [Online]. 5

[CM 17b] CM LABS SIMULATIONS: Vortex Studio. http://www.cm-labs.com/, 2017. [Online]. 2, 5

[Cou17] COUMANS E.: Bullet physics library. http://bulletphysics.org/, 2017. [Online]. 2

[CPS92] COTTLE R. W., PANG J.-S., STONE R. E.: *The Linear Complementarity Problem*. SIAM, 1992. 1

[DF95] DIRKSE S. P., FERRIS M. C.: The path solver: a nommonotone stabilization scheme for mixed complementarity problems. *Optimization Methods and Software 5*, 2 (1995), 123–156. 1

[DS11] DRUMWRIGHT E., SHELL D. A.: An evaluation of methods for modeling contact in multibody simulation. In *IEEE International Conference on Robotics and Automation* (Shanghai, China, June 18–22, 2011), pp. 1695–1701. 2

[DS12] DRUMWRIGHT E., SHELL D. A.: Extensive analysis of linear complementarity problem (LCP) solver performance on randomly generated rigid body contact problems. In *IEEE/RSJ International Conference on Intelligent Robots and Systems* (Vilamoura, Portugal, October 7–12 2012), pp. 5034–5039. 2

[Erl05] ERLEBEN K.: *Stable, robust, and versatile multibody dynamics animation*. PhD thesis, University of Copenhagen, Denmark, 2005. 3

[Erl07] ERLEBEN K.: Velocity-based shock propagation for multibody dynamics animation. *ACM Transactions on Graphics 26*, 2 (2007), 1–20. 2, 3

[Erl13] ERLEBEN K.: Numerical methods for linear complementarity problems in physics-based animation. In *ACM SIGGRAPH Conference on Computer Graphics and Interactive Techniques* (Anaheim, CA, USA, July 21–25, 2013), pp. 1–42. 1

[ETT15] EREZ T., TASSA Y., TODOROV E.: Simulation tools for model-based robotics: Comparison of Bullet, Havok, Mujoco, ODE and Physx. In *IEEE International Conference on Robotics and Automation* (Seattle, WA, USA, May 26–30, 2015), pp. 4397–4404. 2

[GY11] GIOVANNI S., YIN K.: Locotest: Deploying and evaluating physics-based locomotion on multiple simulation platforms. In *Motion in Games* (2011), Allbeck J. M., Faloutsos P., (Eds.), Springer, pp. 227–241. 2

[Hav17] HAVOK: Havok Physics. https://www.havok.com/physics/, 2017. [Online]. 2

[IPPN14] IVALDI S., PETERS J., PADOIS V., NORI F.: Tools for simulating humanoid robot dynamics: A survey based on user feedback. In *IEEE-RAS International Conference on Humanoid Robots* (Madrid, Spain, November 18–20, 2014), pp. 842–849. 2

[JP94] JÚDICE J. J., PIRES F. M.: A block principal pivoting algorithm for large-scale strictly monotone linear complementarity problems. *Computers & Operations Research 21*, 5 (1994), 587–596. 3

[Júd94] JÚDICE J. J.: Algorithms for linear complementarity problems. In *Algorithms for Continuous Optimization*, Spedicato E. G., (Ed.). Springer, 1994, pp. 435–474. 1

[Lac06] LACOURSIÈRE C.: *A regularized time stepper for multibody systems*. Tech. rep., Umeå University, Sweden, HPC2N and Department of Computing Science, 2006. 2

[Lac07] LACOURSIÈRE C.: *Ghosts and machines: regularized variational methods for interactive simulations of multibodies with dry frictional contacts*. PhD thesis, Umeå University, 2007. 1, 3

[LL11] LACOURSIÈRE C., LINDE M.: *Spook: a variational timestepping scheme for rigid multibody systems subject to dry frictional contact*. Tech. rep., Umeå University, Sweden, HPC2N and Department of Computing Science, 2011. 3

[LLWT13] LACOURSIÈRE C., LU Y., WILLIAMS J., TRINKLE J.: Standard interface for data analysis of solvers in multibody dynamics. In *Canadian Conference on Nonlinear Solid Mechanics* (Montréal, QC, Canada, July 23-26, 2013). 2

[LT15] LU Y., TRINKLE J.: Comparison of multibody dynamics solver performance: Synthetic versus realistic data. In *ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference* (Boston, MA, USA, August 2–5, 2015), pp. 1–10. 2

[LWTL14] LU Y., WILLIAMS J., TRINKLE J., LACOURSIERE C.: A framework for problem standardization and algorithm comparison in multibody system. In *ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference* (Buffalo, NY, USA, August 17–20, 2014), pp. 1–10. 2

[MS86] MANGASARIAN O. L., SHIAU T.-H.: Error bounds for monotone linear complementarity problems. *Mathematical Programming 36*, 1 (1986), 81–89. 4

[MY88] MURTY K. G., YU F.-T.: *Linear complementarity, linear and nonlinear programming*. Heldermann Verlag, 1988. 1

[NE15] NIEBE S., ERLEBEN K.: Numerical methods for linear complementarity problems in physics-based animation. In *Synthesis Lectures on Computer Graphics and Animation*, Barsky B. A., (Ed.). Morgan & Claypool Publishers, 2015, pp. 1–159. 1

[NVI17] NVIDIA: PhysX. https://www.geforce.com/hardware/technology/physx, 2017. [Online]. 2

[Pan86] PANG J.-S.: Inexact newton methods for the nonlinear complementarity problem. *Mathematical Programming 36*, 1 (1986), 54–71. 4

[Smi17] SMITH R.: Open dynamics engine. http://www.ode.org/, 2017. [Online]. 2

[SNE10] SILCOWITZ M., NIEBE S., ERLEBEN K.: Interactive rigid body dynamics using a projected gauss–seidel subspace minimization method. In *International Conference on Computer Vision, Imaging and Computer Graphics* (Angers, France, May 17–21 2010), pp. 218–229. 3, 5

[ST96] STEWART D. E., TRINKLE J. C.: An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and coulomb friction. *International Journal for Numerical Methods in Engineering 39*, 15 (1996), 2673–2691. 2

[Tod17] TODOROV E.: Multi-Joint dynamics with Contact (MuJoCo). www.mujoco.org, 2017. [Online]. 2

[WLN*13] WILLIAMS J., LU Y., NIEBE S., ANDERSEN M., ERLEBEN K., TRINKLE J. C.: RPI-MATLAB-Simulator: A tool for efficient research and practical teaching in multibody dynamics. In *Workshop in Virtual Reality Interactions and Physical Simulation* (Lille, France, November 27–29, 2013), pp. 71–80. 2