Preprint ANL/MCS-P1495-0508

*Optimization Methods and Software*
Vol. 00, No. 00, January 2008, 1–23

## RESEARCH ARTICLE

# A computational study of the use of an optimization-based method for simulating large multibody systems

COSMIN PETRA[†], BOGDAN GAVREA[‡], MIHAI ANITESCU [*§] and FLORIAN POTRA[†]

[†]*Department of Mathematics and Statistics, University of Maryland Baltimore County, MD, USA*

[‡]*Faculty of Automation and Computer Science, Department of Mathematics,Technical University of Cluj-Napoca, Romania*

[§]*Mathematics and Computer Science Division, Argonne National Laboratory, IL, USA*

( *May 2008*)

The present work aims at comparing the performance of several quadratic programming (QP) solvers for simulating large-scale frictional rigid-body systems. Traditional time-stepping schemes for simulation of multibody systems are formulated as linear complementarity problems (LCPs) with copositive matrices. Such LCPs are generally solved by means of Lemke-type algorithms and solvers such as the PATH solver proved to be robust. However, for large systems, the PATH solver or any other pivotal algorithm becomes unpractical from a computational point of view. The convex relaxation proposed by one of the authors allows the formulation of the integration step as a quadratic program, for which a wide variety of state-of-the-art solvers are available. In what follows we report the results obtained solving that subproblem when using the QP solvers MOSEK, OOQP, TRON, and BLMVM. OOQP is presented with both the symmetric indefinite solver MA27 and our Cholesky reformulation using the CHOLMOD package. We investigate computational performance and address the correctness of the results from a modeling point of view. We conclude that the OOQP solver, particularly with the CHOLMOD linear algebra solver, has predictable performance and memory use patterns and is far more competitive for these problems than are the other solvers.

**Keywords:** quadratic programming; large scale optimization; rigid multi-body; contact and friction;

**AMS Subject Classification**: 65K10, 90C33, 90C20, 37N15

## 1. Introduction

The dynamic rigid multibody contact problem is concerned with predicting the motion of several rigid bodies in contact and is one of the fundamental paradigms in modern computational science. It appears in the description of fuel motion in the pebble bed reactor [30], in the compaction of nanopowders [13, 37], and in the study of biological membranes [32, 36, 47, 55]. Such simulations are also used extensively in structural engineering [25], pedestrian evacuation dynamics [35], granular matter [45], robotics simulation and design [26], and virtual reality [5].

*Corresponding author. Email: anitescu@mcs.anl.gov

ISSN: 1055-6788 print/ISSN 1029-4937 online

DOI: 10.1080/1055678xxxxxxxxxxxx
http://www.informaworld.com

Approaches used in the past for the numerical approximation of rigid multibody dynamics with contact and friction include piecewise DAE approaches [34], acceleration-force linear complementarity problem (LCP) approaches [12, 43, 54], penalty approaches [21, 42, 48, 49], and velocity-impulse LCP-based time-stepping methods [9, 10, 44, 50, 51]. LCP-based time-stepping schemes for simulating multibody systems are formulated as LCPs with copositive matrices. Such LCPs are generally solved by means of Lemke-type algorithms, and solvers such as the PATH solver [20, 27] have proved to be robust. For large systems (beyond a few thousand contacts), however, the PATH solver or any other pivotal algorithm becomes impractical from a computational point of view [8, 53].

The computational difficulties associated with the standard frictional LCP formulation cannot be avoided even for small friction coefficients. In this context, in [7], a simple example is used to show that the solution set of the underlying LCP fails to be convex for any nonzero friction coefficients and therefore no polynomial time algorithms are known to exist.

When solving large-scale multibody dynamics, a relaxation of the standard (copositive) LCP formulation is desirable. The convex relaxation introduced in [3] is convergent in the same weak sense as the original, nonconvex scheme [50]. This relaxation formulates the integration step as a convex quadratic program (QP) for which state–of–the–art solvers are available.

In this paper we investigate the performance of several solvers for QP problems that appear from the relaxed formulation of multirigid body dynamics with contact and friction. The examples are obtained by simulating granular flow motion in a system similar to the pebble-bed reactor described in [30]. A description of the simulated multibody system is given in Section 2.1.

We note that QP approaches are popular in the graphics community for simulating rigid-body dynamics with contact and friction [24, 33, 40]. Their most common instance insofar as friction is concerned is probably the box friction model approach. There, the normal force is prescribed either directly or following a frictionless presolve, which is also a convex QP [33]. Variants of this approach are used as an option in many major physics engines for games, such as ODE (Open Dynamics Engine), OpenTissue, and Vortex (some discussion in [15]). All the QPs appearing in such approaches have a sparsity structure with an incidence graph closely related to the graph with nodes in the center of the bodies and edges between bodies in potential contact. The sparsity we encounter here is based on the same graph. The number of bodies (thousands) and contacts (tens of thousand) we consider is comparable or larger to the one considered in most of the applications targeted by such physics engines [24]. We therefore expect that our findings will extend to those applications as well. We point out, however, that to our knowledge the method introduced in [3] is the only optimization-based method for rigid-body dynamics that is provably convergent, at least in a weak sense [50], to a continuous-time solution for any friction coefficient. This observation motivates our choice of subject for the computational experiment.

The paper has two main parts. In Section 2 we introduce the optimization problems to be solved. In this context we specialize the QP–formulation of [3] to the application described in Section 2.1 and address the modeling of the pebble bed reactor application. We plan to exploit two QP formulations: the primal QP formulation and the dual QP formulation, which takes the form of a bound constrained minimization problem. For these formulations, four QP-solvers are used to compare computational efficiency and to analyze ensemble properties of the simulated trajectories. We use two interior-point solvers, MOSEK [1] and OOQP [29] and two projected gradients solvers (on the dual formulation), TRON [39] and BLMVM

[14]. A comparison between these solvers is given in Section 3, details of the numerical experiments are given in Section 4, and a summary of our observations is given in Section 5.

## 2. Quadratic Programming Subproblems of Time-Stepping Methods

We now discuss the origin and structure of the QP subproblems in the time-stepping approach in [3]. We will apply the formulation to the problem of the simulation of the granular flow of the fuel pebbles in a pebble-bed reactor (PBR) [30]. This example problem has the advantage that we can easily scale it up in the number of the rigid bodies and contacts, making it ideal for extrapolating the behavior of the solvers for increasing dimension of the problem.

We present an image of the reactor vessel in Figure 1, with all pebbles at rest. The rigid bodies – the pebbles – are tennis-ball-sized and contain uranium oxide. They are extracted from the bottom of the vessel and reinserted through the top [30]. When fully loaded, the reactor has about 400,000 such bodies, but important assessments can be extracted from simulations involving a smaller number [46]. The reactor vessel in which the motion of the pebbles is simulated is composed of a truncated cone and a cylinder that is opened at both ends; see Figure 1. For modeling purposes, we index the cylindrical surface of the vat by $-3$, the lateral surface of the truncated cone by $-2$, and the bottom of the vat by $-1$.

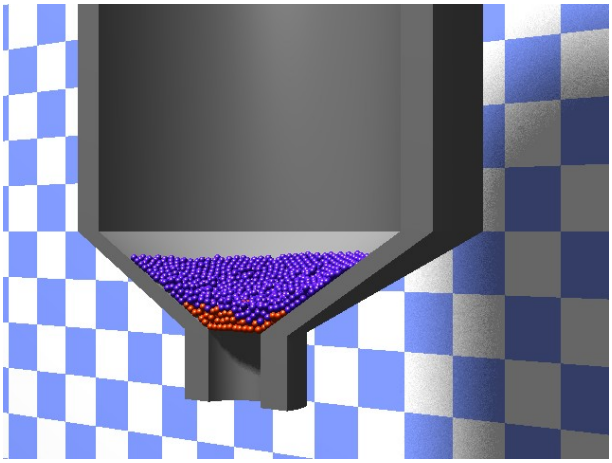


Figure 1. Cross-section of the reactor vessel with 3200 pebbles at the end of the simulation. The pebbles are colored based on the originating position.

Most of the following modeling steps apply to general rigid multibody dynamics. However, some topics particular to the PBR simulation will be pointed out as they occur.

### 2.1. *The Model*

In this section we describe the various modeling steps that convert a rigid multibody dynamics problem with contact and friction to a sequence of quadratic programs such as (2.10). The model is based on the convex relaxation introduced in [3]. The merits and shortcomings of that relaxation, insofar accuracy of predicting frictional behavior is concerned, are presented in [3] and are not the subject of this work. Here, we focus on the setup and resolution of (2.10).

Assume that the rigid-body system is composed of $N$ bodies. The position of body $i$ is $q^{(i)} = (x_i, y_i, z_i, \theta_i, \alpha_i, \gamma_i)^T$, $i = 0, ..., N-1$, where the first three com-

ponents are the Cartesian coordinates of the center (in a fixed inertial frame) and the last three represent the orientation of a reference point $P^{(i)}$ on the sphere. The position $q$ of the entire system is obtained by concatenating the positions of each body, namely, $q = \left(q^{(0)^T}, q^{(2)^T}, ..., q^{(N-1)^T}\right)^T$. In a similar fashion, we define the generalized velocity of the system to be $v \in \boldsymbol{R}^{6N}$, $v = \Gamma(q)\dfrac{dq}{dt}$. Here $\Gamma(q)$ is a smooth mapping that converts the derivatives of the position coordinates to the generalized velocities [34]. For rigid bodies, as opposed to material points, $\Gamma(q)$ is different from the identity [34].

**Nonpenetration constraints.** For bodies $i$, and $j$, we assume that we have signed gap functions functions $\Phi^{(i,j)}(q)$, such that

$$\Phi^{(i,j)}(q) = \begin{cases} < 0 \text{ the two bodies penetrate,} \\ = 0 \text{ the two bodies are touching,} \\ > 0 \text{ the two bodies are separated.} \end{cases}$$

In this case, the nonpenetration constraints simply become

$$\Phi^{(i,j)}(q) \geq 0.$$

For PBR, the nonpenetration constraints are imposed between *pebble–pebble* and *pebble–wall* interaction. The constraint indices are $(i, j)$, for $(i \in \{-3, ..., (N-1)\}$, $j \in \{0, ..., (N-1)\}$ and $j > i)$. In addition, the signed gap functions are particularly easy to define. For example, for two spheres of radius $R$, and with centers of mass at positions $x_1, y_1, z_1$ and $x_2, y_2, z_2$, a signed gap function is

$$\Phi^{(i,j)} = (x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2 - 4R^2.$$

The pebble-wall gap functions use the distance from a point to a simple surface and are also immediate to define.

**Contact specifications.** For each contact we define the normal and tangential directions in generalized coordinates as follows. Consider the generic contact depicted in Figure 2. Let $\overrightarrow{n}$ denote the unit outward normal for the horizontally aligned body. Let $C_1$ and $C_2$ be the centers of mass for each body, and let $\overrightarrow{r}_1$ and $\overrightarrow{r}_2$ be the position vectors of the contact point relative to $C_1$ and $C_2$ respectively (all vectors are represented in world frame coordinates). Then the 3–dimensional vector $\overrightarrow{n}$ is mapped in generalized coordinates into the 12–dimensional vector $n$, defined as follows

$$\overrightarrow{n} \longmapsto n := \begin{pmatrix} \overrightarrow{n} \\ \overrightarrow{r}_1 \times \overrightarrow{n} \\ -\overrightarrow{n} \\ -\overrightarrow{r}_2 \times \overrightarrow{n} \end{pmatrix}. \tag{2.1}$$

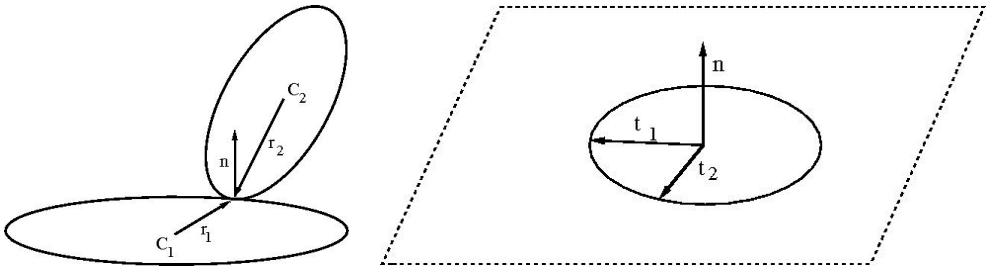
In a similar fashion one obtains the generalized coordinates vectors $t_1$ and $t_2$ from $\overrightarrow{t}_1$ and $\overrightarrow{t}_2$, respectively.

The Coulomb friction model prescribes that the tangential force is inside a disk proportional to the one in Figure 2. To allow for the use of a quadratic programming approach (or an LCP in the unrelaxed case), we use a polygonal approximation of the friction disk in Figure 2(b) [9, 51]. The disk is approximated by an inscribed polygon, whose quality depends on the number $p$ of tangent directions used.

We define those $p$ tangent vectors, $\overrightarrow{d}_s$, by

$$\overrightarrow{d}_s := \cos\left(\frac{2\pi s}{p}\right)\overrightarrow{t}_1 + \sin\left(\frac{2\pi s}{p}\right)\overrightarrow{t}_2, \quad s = 1, ..., p. \tag{2.2}$$

Clearly the vectors $\overrightarrow{d}_s$ are direction vectors in $\boldsymbol{R}^3$, and any point in $span\{\overrightarrow{d}_1, ..., \overrightarrow{d}_p\}$ can be written as a nonnegative linear combination of these vectors. It is easy to see that the generalized coordinate version of the directions




(a) (b)

Figure 2. (a) Generic contact between two bodies, (b) Tangent space at contact.

in (2.2) are obtained in the same fashion, namely,

$$d_s := \cos\left(\frac{2\pi s}{p}\right)t_1 + \sin\left(\frac{2\pi s}{p}\right)t_2, \quad s = 1, ..., p. \tag{2.3}$$

Given that, for a system of $N$ bodies the configuration space (the space where $q$ lives) has dimension $6N$, we embed the contact data in $\boldsymbol{R}^{6N}$. More precisely, assume that the $k$-th indexed contact is established between body $i$ ($i \geq 0$) and body $j$ ($j > i$). The normal direction in the $6N$ dimensional space is then given by

$$n^{(k)} = \left(O_{1,6(i-1)}, \overrightarrow{n}_k^T, (\overrightarrow{r}_i \times \overrightarrow{n}_k)^T, O_{1,6(j-i-1)}, -\overrightarrow{n}_k^T, (-\overrightarrow{r}_j \times \overrightarrow{n}_k)^T, O_{1,6(N-j)}\right)^T, \tag{2.4}$$

where $O_{1,\alpha}$ represents a zero row vector of length $\alpha$, whenever $\alpha > 0$ and the empty vector otherwise. Here $n_k$ is the three-dimensional normal vector at the contact $k$.

In the case of PBR, for a *pebble–wall* interaction $(i, j)$, $i < 0$, only the second nonzero block will contribute to $n^{(k)}$.

In the same fashion one defines the tangential directions $d_s^{(k)} \in \boldsymbol{R}^{6N}$, $s = 1, ..., p_k$, namely,

$$d_s^{(k)} = \left(O_{1,6(i-1)}, \overrightarrow{d}_{sk}^T, (\overrightarrow{r}_i \times \overrightarrow{d}_{sk})^T, O_{1,6(j-i-1)}, -\overrightarrow{d}_{sk}^T, (-\overrightarrow{r}_j \times \overrightarrow{d}_{sk})^T, O_{1,6(N-j)}\right)^T. \tag{2.5}$$

In (2.5), $d_{sk}$, $s = 1, ..., p_k$ are the $\boldsymbol{R}^{p_k}$ direction vectors used in the approximation of the friction disk at contact $k$. The matrix associated with the polyhedral approximation of the friction disk at contact $(k)$ is the matrix having its columns the directions $d_s^{(k)}$, i.e., $D^{(k)} \in \boldsymbol{R}^{6N \times p_k}$, $D^{(k)} = \left(d_1^{(k)}, ..., d_{p_k}^{(k)}\right)$, where $p_k$ represents the number of friction generators for contact $(k)$.

In the formulation of the integration step we will deal with matrices $\widehat{D}^{(k)} \in \boldsymbol{R}^{6N \times p_k}$ of the form

$$\widehat{D}^{(k)} = \left(n^{(k)} + \mu d_1^{(k)}, ..., n^{(k)} + \mu d_{p_k}^{(k)}\right), \tag{2.6}$$

where $\mu$, $\mu \in (0,1]$ is the friction coefficient, which is assumed to be the same for all contacts . We are interested in the maximal value of $p_k$ ($p_k < 6N$) for which the matrix $\widehat{D}^{(k)}$ has full column rank, $rank(\widehat{D}^{(k)}) = p_k$. It is easy to see that for $p_k = 3$, $rank(\widehat{D}^{(k)}) = p_k$, while for $p_k > 3$ we obtain a rank-deficient matrix. This observation will be used when formulating the integration timestep as a bound–constrained minimization problem (this is what we call the dual formulation). For the PBR simulation we choose $p_k = 3$ for all contacts $k$.

**External and inertial forces.** We denote by $M \in \boldsymbol{R}^{6N\times 6N}$ the generalized mass matrix of the system. In general, in a fixed coordinate frame, this matrix depends explicitly on the position $q$ , i.e., $M := M(q)$. We also denote by $k_{app}$ the sum between the external forces and the inertial forces. The inertial forces involves derivatives of $M(q)$ [34].

For the PBR example, we are dealing only with spherical bodies. The generalized mass matrix is diagonal with positive entries and constant with respect to $q$ [34]. Since the mass matrix is constant, the inertial forces are zero. This implies that, besides contact forces, the only forces acting on the system are external forces.

For PBR, because of the heaviness of uranium, we can ignore the effect of the cooling flow over the dynamics [30]. Therefore, we can assume that only gravitational forces are acting, and the (noncontact) applied forces $k_{app} \in \boldsymbol{R}^{6N}$ can be written as

$$k_{app} = \left(u^T, ..., u^T\right)^T.$$

Here $u \in \boldsymbol{R}^6$, $u = (0, 0, -g, 0, 0, 0)^T$, for some positive constant $g$.

**Linearized active contacts constraints.** Given a current position $q$ of the system, we compute the set of active contacts by using the signed distance functions $\Phi^{(i,j)}(q)$. Given a positive scalar $\epsilon$, a contact $(k)$ corresponding to the rigid-body pair $(i, j)$ is considered active if

$$\Phi^{(k)}(q) = \Phi^{(i,j)}(q) \le \epsilon.$$

For a given configuration $q$, we denote by $\mathcal{A}(q, \epsilon)$ the set of all active contacts. More precisely,

$$\mathcal{A}(q,\epsilon) = \left\{k \in \mathbb{Z}_+^2 \mid (k) = (i,j),\ \Phi^{(i,j)}(q) \le \epsilon\right\}. \tag{2.7}$$

If $(k) \notin \mathcal{A}(q, \epsilon)$, the corresponding nonpenetration constraint is simply ignored by the QP (2.10). If $(k) \in \mathcal{A}(q, \epsilon)$, then the nonpenetration constraint $\Phi^{(i,j)} > 0$ is enforced at time $t^l$ and position $q^l$ by

$$\left(n^{(k)}(q^l)\right)^T v + \mu \left(d_s^{(k)}(q^l)\right)^T v \ge -\frac{1}{h}\Phi^{(k)}(q^l), \quad s = 1, 2, \ldots, p_k. \tag{2.8}$$

Here $h$ is the timestep of the scheme. It does not need to be constant for stability [6], but we choose it so for graphical simplicity. The first and last terms in this equation are the linearization of the nonpenetration constraint. The last term is stable as $h \to 0$ [3, 6]. The middle term is unique to the scheme in [3]. Its physical significance is based on a microscopic realization of surface asperities that result in macroscopic friction coefficient $\mu$.

It has been shown that the value of $\epsilon$ does not affect the convergence as $h \to 0$ [3, 6]. But its choice has important practical consequences. A value that is too large results in an exceedingly large QP. The QP is still consistent but may be

computationally expensive. A value that is too small may result in too many non-penetration constraints being dropped and in excessive penetration at time step $l+1$. An appropriate value for $\epsilon$ should be comparable to the product between the maximum velocity in the system and the timestep.

**Newton's second law.** The discretized version of Newton's second law is written at the velocity-impulse level as

$$M\left(v^{l+1}-v^{l}\right)-z^{l+1}=hk_{app}, \tag{2.9}$$

where $hk_{app}$ are the external impulses and $z^{l+1}$ represent the contact impulses (normal and tangential/frictional contact impulses):

$$z^{l+1}=\sum_{k\in\mathcal{A}(q^{l},\epsilon)}\sum_{s=1}^{p_k}\beta_s^{(k)}\left(n^{(k)}(q^{l})+\mu d_s^{(k)}(q^{l})\right).$$

Here $M$ is the mass matrix, and each vector of multipliers satisfies $\beta^{(k)}=\left(\beta_1^{(k)},\beta_2^{(k)},\ldots,\beta_{p_k}^{(k)}\right)\in\boldsymbol{R}^{p_k}$ and $\beta^{(k)}\geq 0$.

## 2.2. *The Integration Step*

In what follows we present the formulation proposed in [3] and its dual. Here, we consider only the case of totally plastic collisions. The scheme can be modified to accommodate partially elastic or totally elastic collisions by means of a restitution coefficient [2]. We point out, however, that the issue of predictive modeling of simultaneous nonplastic collisions is far from being settled in rigid body dynamics [16].

### *2.2.1. Primal Formulation*

Let $h>0$ denote the size of the integration timestep. We denote by $q^l$ the position and by $v^l$ the velocity of the system at time $t_l=lh$. In the optimization-based time-stepping scheme introduced in [3], the new velocity $v^{l+1}$ is obtained by solving the following quadratic problem:

$$\begin{aligned}&\min\ \tfrac{1}{2}v^{T}Mv+\left(f^{l}\right)^{T}v\\&\text{s.t.}\ \ \left(n^{(k)}(q^{l})\right)^{T}v+\mu\left(d_s^{(k)}(q^{l})\right)^{T}v\geq-\tfrac{1}{h}\Phi^{(k)}(q^{l})\\&\qquad k\in\mathcal{A}(q^{l},\epsilon),\ s=1,2,\ldots,p_k\end{aligned} \tag{2.10}$$

In (2.10), $f^l$ is obtained by the following formula:

$$f^{(l)}=-v^{l}-hk_{app}. \tag{2.11}$$

Equations (2.9) and (2.8) are satisfied as part of the optimality conditions for (2.10).

### *2.2.2. Dual Formulation*

The dual formulation can be obtained by standard duality techniques. In our case, we assign the Lagrange multipliers $\lambda$ to the constraints in (2.10). Then, we write the optimality conditions for (2.10), and we eliminate $v$ using the positive definiteness of $M$. This procedure results in the dual QP program (2.13).

Let us denote by $A^l$ and $b^l$ the matrix and the right-hand side, respectively, of the inequality constraints in (2.10). In terms of the notation introduced in (2.6), the matrix $A^l$ has the form

$$A^l = \begin{pmatrix} \left(\widehat{D}^{k_1}(q^l)\right)^T \\ \vdots \\ \left(\widehat{D}^{k_p}(q^l)\right)^T \end{pmatrix}, \tag{2.12}$$

where the active set $\mathcal{A}(q^l, \epsilon) = \{k_i \mid i = 1, ...p\}$. The vector $b^l$ is composed of block vectors in $\mathbb{R}^3$, with the block corresponding to contact $k_i$ having all its components equal to $-\frac{1}{h}\Phi^{(k_i)}(q^l)$. In the notation described above, the dual problem takes the form

$$\begin{array}{l} \min \ \frac{1}{2}\lambda^T P^l \lambda + \left(\kappa^l\right)^T \lambda, \\ \text{s.t.} \ \ \lambda \geq 0 \end{array} \tag{2.13}$$

where $P^l = A^l M^{-1} \left(A^l\right)^T$ and $\kappa^l = -b^l - A^l f^l$.

The dual formulation (2.13) is a bound-constrained quadratic programming problem. Therefore, for solving it, we can use not only general-purpose quadratic programming algorithms, such as interior points, but also iterative algorithms of the projected gradient type. Therefore, it is a good formulation for benchmarking the performance of various solvers for quadratic programming.

### 2.3. *Pointed friction cone and duality*

Consider the $\epsilon$-active set $\mathcal{A}(q^l, \epsilon)$, which is obtained from (2.7) for the current configuration $q^l$. For $k \in \mathcal{A}(q^l, \epsilon)$ and let $\widetilde{D}^{(k)}$ denote the matrix of generalized tangential directions, namely,

$$\widetilde{D}^{(k)}(q^l) := \widetilde{D}^{(k)} = \left(d_1^{(k)}, d_2^{(k)}, \ldots, d_{p_k}^{(k)}\right), \tag{2.14}$$

where the generalized tangential directions $d_s^{(k)} := d_s^{(k)}(q^l), s = 1, 2, \ldots, p_k$ are given in (2.3). We now define the $\epsilon-$ active friction cone $\mathcal{FC}(q^l, \epsilon)$ by

$$\mathcal{FC}(q^l, \epsilon) = \left\{ \sum_{k \in \mathcal{A}(q^l, \epsilon)} \widetilde{D}^{(k)} \beta^{(k)} \ \Big| \ \beta^{(k)} \in \mathbb{R}^{p_k}, \ \beta^{(k)} \geq 0, \ \right\}. \tag{2.15}$$

We say that the friction cone is **pointed** if the following implication holds:

$$z = \sum_{k \in \mathcal{A}(q^l, \epsilon)} \widetilde{D}^{(k)} \beta^{(k)} \in \mathcal{FC}(q^l, \epsilon), \ z = 0 \ \Rightarrow \ \beta^{(k)} = 0, \forall \ k \in \mathcal{A}(q^l, \epsilon). \tag{2.16}$$

In other words $\mathcal{FC}(q^l, \epsilon)$ is pointed if it doesn't contain any proper subspaces.

It has been shown in [8] that pointedness of the friction cone implies that the Mangasarian-Fromowitz constraint qualification (MFCQ) holds for the convex program (2.10). Whenever MFCQ holds the multipliers $\lambda$ in (2.13) are bounded [28]. Therefore, pointedness of the friction cone together with the existence of a feasible point for (2.10) guarantees no duality gap. This result is essential when comparing

QP solvers that use either the primal or the dual formulation. Loss of the pointedness regularity assumption corresponds, from a physical point of view, to jamming [4], a phenomenon that does not occur in our simulations.

For isolated QPs we can compare the correctness of the results obtained from solving the two different formulations, by either measuring the duality gap or by passing from a dual solution to its primal correspondent. When running an entire simulation, however, the use of different solvers will likely cause totally different individual configurations. This is motivated by the fact that, by nature, the system is chaotic. Therefore, to measure the correctness of an entire simulation, at least partially, we use ensemble properties, such as the kinetic energy of the entire system.

## 3. Algorithms and Software Packages Used

We now describe the properties of several packages used to solve (2.13). We use two types of packages. The first, of the interior-point type, OOQP and MOSEK, solve the primal-dual formulation of both (2.13) and (2.10). For the OOQP solver, we use two formulations, one of which involves our adaptation of the CHOLMOD linear algebra package for use with OOQP. The second, of the projected gradient type, TRON and BLVM, solve the dual problem (2.13).

### 3.1. *OOQP*

OOQP (Object-Oriented software for Quadratic Programming) is a C++ package for solving convex quadratic programming problems. It is based on primal-dual interior-point methods and can be used to solve a variety of forms of quadratic problems such as general sparse QPs, QPs with "box" constraints, QPs coming from support vector machines, and Huber regression problems. Its object-oriented design allows easy adaptation for specialized QP formulations or use of new linear algebra solvers.

In our experiments OOQP's general sparse formulation is used to solve the primal form (2.10). Two distinct linear algebra packages are used: MA27 for which an interface is included in OOQP distribution and CHOLMOD for which we developed an interface and reformulated the linear systems. We denote the two versions of OOQP by OOQP-MA27 and OOQP-CHOL, respectively.

#### *3.1.1. OOQP general sparse formulation*

In OOQP the convex quadratic problem subject to linear constraints is considered in the following general form:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}x^T Q x + c^T x \\ \text{subj to:} & Ax = b \\ & c_l \leq Cx \leq c_u \\ & x_l \leq x \leq x_u, \end{array} \tag{3.17}$$

where $Q \in \boldsymbol{R}^{n\times n}$ , $c \in \boldsymbol{R}^n$, $A \in \boldsymbol{R}^{m_y\times n}$, $C \in \boldsymbol{R}^{m_z\times n}$, $c_l, c_u \in \boldsymbol{R}^{m_z}$, and $x_l, x_u \in \boldsymbol{R}^n$.

It is well known that at each iteration of the interior-point methods one or more linear systems need to be solved. In what follows, we describe the way OOQP manages the KKT conditions and builds the linear systems.

The Lagrangian function corresponding to the quadratic problem (QP) (3.17) is $L(x) = \frac{1}{2}x^T Q x + c^T x + y^T(b - Ax) + \lambda^T(c_l - Cx) + \pi^T(Cx - c_u) + \gamma^T(x_l - x) +$

$\phi^T(x - x_u)$; hence, the KKT conditions can be written as follows:

$$\begin{aligned} Qx - A^T y - C^T\lambda + C^T\pi - \gamma + \phi + c = 0 \\ Ax = b \\ 0 \le Cx - c_l = t \perp \lambda \ge 0 \\ 0 \le c_u - Cx = u \perp \pi \ge 0 \\ 0 \le x - x_l = v \perp \gamma \ge 0 \\ 0 \le x_u - x = w \perp \phi \ge 0. \end{aligned}$$

The interior-point iterations consist of solving the perturbed KKT systems

$$F(x,y,t,u,v,w,\lambda,\pi,\gamma,\phi) = \begin{bmatrix} Qx - A^T y - C^T\lambda + C^T\pi - \gamma + \phi + c \\ -Ax + b \\ Cx - t - c_l \\ -Cx - u + c_u \\ x - v - x_l \\ -x - w + x_u \\ \lambda t - \mu_k e \\ \pi u - \mu_k e \\ \gamma v - \mu_k e \\ \phi w - \mu_k e \end{bmatrix} = 0,$$

for a sequence of positive $\{\mu_k\}$ that converges to zero, while maintaining the positiveness of $t, u, v, w, \lambda, \pi, \gamma, \phi$. The Newton's method is used to (approximately) solve the above nonlinear system, so the linear algebra consists of solving linear systems of form $F'\Delta d = -F$, where the Jacobian $F'$ is given by

$$F' = \begin{bmatrix} Q & -A^T & 0 & 0 & 0 & 0 & -C^T & C^T & -I & I \\ -A & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ C & 0 & -I & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -C & 0 & 0 & -I & 0 & 0 & 0 & 0 & 0 & 0 \\ I & 0 & 0 & 0 & -I & 0 & 0 & 0 & 0 & 0 \\ -I & 0 & 0 & 0 & 0 & -I & 0 & 0 & 0 & 0 \\ 0 & 0 & \Lambda & 0 & 0 & 0 & T & 0 & 0 & 0 \\ 0 & 0 & 0 & \Pi & 0 & 0 & 0 & U & 0 & 0 \\ 0 & 0 & 0 & 0 & \Gamma & 0 & 0 & 0 & V & 0 \\ 0 & 0 & 0 & 0 & 0 & \Phi & 0 & 0 & 0 & W \end{bmatrix},$$

with $\Lambda = \mathrm{diag}(\lambda)$, $\Pi = \mathrm{diag}(\pi)$, $\Gamma = \mathrm{diag}(\gamma)$, $\Phi = \mathrm{diag}(\phi)$, $T = \mathrm{diag}(t)$, $U = \mathrm{diag}(u)$, $V = \mathrm{diag}(v)$, and $W = \mathrm{diag}(w)$.

The remaining of this section goes into the details of solving linear system $F'\Delta d = -F$. Using the expression of $F'$, we can write the equivalent form

$$\begin{cases} Q\Delta x - A^T\Delta y - C^T\Delta\lambda + C^T\Delta\pi - \Delta\gamma + \Delta\phi = r_Q \\ A\Delta x = r_y \\ C\Delta x - \Delta t = r_t \\ -C\Delta x - \Delta u = r_u \\ \Delta x - \Delta v = r_v \\ -\Delta x - \Delta w = r_w \\ \Lambda\Delta t + T\Delta\lambda = r_\lambda \\ \Pi\Delta u + U\Delta\pi = r_\pi \\ \Gamma\Delta v + V\Delta\gamma = r_\gamma \\ \Phi\Delta w + W\Delta\phi = r_\phi. \end{cases} \tag{3.18}$$

The exact form $-F$ of the right-hand side is not of interest in this discussion of the linear algebra layer since it is specific to the interior-point algorithm. We denoted the right-hand side vectors by $r_Q, r_y, r_t$, $r_u, r_v, r_w, r_\lambda, r_\pi, r_\gamma$, and $r_\phi$.

Using equations 5, 6, 9, and 10 from (3.18), we can express $\Delta\gamma - \Delta\phi = -V^{-1}\Gamma\Delta v + W^{-1}\Phi\Delta w + V^{-1}r_\gamma - W^{-1}r_\phi = -V^{-1}\Gamma(\Delta x - r_v) + W^{-1}\Phi(-\Delta x - r_w) + V^{-1}r_\gamma - W^{-1}r_\phi$. And if we denote $D_1 = V^{-1}\Gamma + W^{-1}\Phi$, then

$$\Delta\gamma - \Delta\phi = -D_1\Delta x + (V^{-1}r_\gamma - W^{-1}r_\phi + V^{-1}\Gamma r_v - W^{-1}\Phi r_w). \tag{3.19}$$

Let $\Delta z = \Delta\lambda - \Delta\pi$. From the sixth equation we get $\Delta\lambda = -T^{-1}\Lambda\Delta t + T^{-1}r_\lambda$ and from the seventh equation $\Delta\pi = -U^{-1}\Pi\Delta u + U^{-1}r_\pi$. We also use the third and fourth equation to get $\Delta z = -T^{-1}\Lambda(C\Delta x - r_t) + U^{-1}\Pi(-C\Delta x - r_u) + T^{-1}r_\lambda - U^{-1}r_\pi = -(T^{-1}\Lambda + U^{-1}\Pi)C\Delta x + T^{-1}r_\lambda - U^{-1}r_\pi + T^{-1}\Lambda r_t - U^{-1}\Pi r_u$. Therefore we can write

$$C\Delta x + D_2^{-1}\Delta z = D_2^{-1}(T^{-1}r_\lambda - U^{-1}r_\pi + T^{-1}\Lambda r_t - U^{-1}\Pi r_u), \tag{3.20}$$

where $D_2 = T^{-1}\Lambda + U^{-1}\Pi$.

Both $D_2^{-1}$ and $D_1^{-1}$ exist, since $T^{-1}\Lambda + U^{-1}\Pi$ and $V^{-1}\Gamma + W^{-1}\Phi$ are diagonal matrices with strictly positive diagonal entries. We substitute (3.19) in the first equation of (3.18) and use (3.20) and the second equation from (3.18) to obtain

$$\begin{cases} (Q + D_1)\Delta x - A^T\Delta y - C^T\Delta z = \overline{r}_Q \\ A\Delta x = \overline{r}_y \\ C\Delta x + D_2^{-1}\Delta z = \overline{r}_z \end{cases} \tag{3.21}$$

where $\overline{r}_Q = r_Q + (V^{-1}r_\gamma - W^{-1}r_\phi + V^{-1}\Gamma r_v - W^{-1}\Phi r_w)$, $\overline{r}_y = r_y$ and $\overline{r}_z = D_2^{-1}(T^{-1}r_\lambda - U^{-1}r_\pi + T^{-1}\Lambda r_t - U^{-1}\Pi r_u$.

Once the solution $(\Delta x, \Delta y, \Delta z)$ of (3.21) is found, the unknowns $\Delta t, \Delta u, \Delta v, \Delta w, \Delta\lambda, \Delta\pi, \Delta\gamma$, and $\Delta\phi$ can be computed by using only diagonal matrix-vector products and vector-vector additions as follows:

$$\begin{cases} \Delta t = C\Delta x + r_t \\ \Delta u = -C\Delta x + r_u \\ \Delta v = \Delta x - r_v \\ \Delta w = -\Delta x - r_w \\ \Delta\lambda = T^{-1}(r_\lambda - \Lambda\Delta t) \\ \Delta\pi = U^{-1}(r_\pi - \Pi\Delta u) \\ \Delta\gamma = V^{-1}(r_\gamma - \Gamma\Delta v) \\ \Delta\phi = W^{-1}(r_\phi - \Phi\Delta w). \end{cases}$$

OOQP solves the symmetric system (3.22) obtained by performing the substitution $(\Delta x, \Delta y, \Delta z) = (\Delta\tilde{x}, -\Delta\tilde{y}, -\Delta\tilde{z})$ in (3.21):

$$\begin{bmatrix} Q + D_1 & A^T & C^T \\ A & 0 & 0 \\ C & 0 & -D_2^{-1} \end{bmatrix} \begin{bmatrix} \Delta\tilde{x} \\ \Delta\tilde{y} \\ \Delta\tilde{z} \end{bmatrix} = \begin{bmatrix} \tilde{r}_x \\ \tilde{r}_y \\ \tilde{r}_z \end{bmatrix}. \tag{3.22}$$

### *3.1.2. About OOQP-MA27*

The linear system (3.22) is known in the interior-point community as the augmented system. OOQP's linear algebra layer for sparse general convex quadratic problems solves the augmented system by using a sparse symmetric indefinite linear solver. The sparse, symmetric, indefinite linear systems are solved by using a

Bunch-Parlett factorization for a matrix $A$. Such a factorization produces permutation matrices $P$, lower triangular matrix $L$, and the block diagonal matrix $D$ with nonsingular $1 \times 1$ and $2 \times 2$ blocks that satisfy $PAP^T = LDL^T$. They are applied to the linear system (3.22).

The OOQP distribution contains interfaces to MA27 [23] and to the newer MA57 [22] linear solvers contained in Harwell Subroutine Library (HSL). We use MA27 because the MA57 solver is not available (free) for U.S. academics. The HSL code MA27 is a collection of FORTRAN routines for solving sparse systems of linear equations by a variant of Gauss elimination. The code and more documentation can be found at http://hsl.rl.ac.uk/archive/hslarchive/packages/packages.html.

### *3.1.3. About our implementation OOQP-CHOL*

In what follows we present the implementation of a new linear algebra layer in OOQP. As we mentioned, the OOQP's default linear algebra layer solve the indefinite symmetric system (3.22). One may take advantage of the particular structure of this system and perform further block elimination. A simple algebraic manipulation of the third equation in (3.22) reveals

$$\Delta\tilde{z} = D_2 C \Delta\tilde{x} - D_2\tilde{r}_z. \tag{3.23}$$

By substituting (3.23) in the first equation of (3.22), we reduced the linear system to

$$\begin{cases} (Q + C^T D_2 C + D_1)\Delta\tilde{x} + A^T\Delta\tilde{y} = \tilde{r}_x + C^T D_2\tilde{r}_z \\ A\Delta x = \tilde{r}_y. \end{cases} \tag{3.24}$$

We express $\Delta\tilde{x}$ in terms of $\Delta\tilde{y}$ by multiplying the first equation of (3.24) with the inverse of $(Q + C^T D_2 C + D_1)$. Notice that $M_1 := (Q + C^T D_2 C + D_1)$ is always invertible and symmetric positive definite since $Q$ and $C^T D_2 C$ are symmetric positive semidefinite and $D_1$ is a diagonal matrix with strictly positive diagonal entries. Using the new expression of $\Delta\tilde{x}$, we rewrite the second equation of (3.24) as

$$-AM_1^{-1}A^T\Delta\tilde{y} + AM_1^{-1}(\tilde{r}_x + C^T D_2\tilde{r}_z) = \tilde{r}_y.$$

We denote $M_2 = AM_1^{-1}A^T$ and obtain the following expression for $\Delta\tilde{y}$

$$M_2\Delta\tilde{y} = AM_1^{-1}(\tilde{r}_x + C^T D_2\tilde{r}_z) - \tilde{r}_y. \tag{3.25}$$

Once $\Delta\tilde{y}$ is known, $\Delta\tilde{x}$ can be obtained from

$$M_1\Delta\tilde{x} = -A^T\Delta\tilde{y} + \tilde{r}_x + C^T D_2\tilde{r}_z. \tag{3.26}$$

As mentioned, $M_1$ is a symmetric positive definite matrix. This implies that $M_2 = AM_1^{-1}A^T$ is also symmetric positive definite. Hence, the use of a Cholesky-based linear solver for solving (3.25) and (3.26) is an appropriate choice.

Performance, reliability, and availability were the main aspects we considered while choosing a sparse direct solver for symmetric positive definite linear systems of equations. We chose CHOLMOD 1.4 [17–19, 56] based on the evaluations from [31] and its ready availability.

In what follows, we describe how our new linear algebra layer within OOQP manages to solve (3.26) and (3.25) for $\Delta\tilde{x}$ and $\Delta\tilde{y}$, respectively. The two interior-point methods implemented in OOQP, Mehrotra and Gondzio, use one matrix

factorization per iteration and at least two backsolves. In other words, at least two linear systems having the same system matrix have to be solved at each iteration of the interior-point algorithm. Therefore, any factorization and other work that is not dependent on the right-hand side must be performed once in the so-called factorization phase. Any other right-hand-side dependent operation is accomplished in the solve phase.

At any iteration of the interior-point method, the factorization phase consists of

- Cholesky factorization $M_1 = L_1 L_1^T$;
- computing $X = L_1^{-1} A^T$ by performing $m_y$ backsolves: $L_1 X = A^T$;
- computing $M_2 = X^T X$;
- Cholesky factorization $M_2 = L_2 L_2^T$.

The solve phase computes $\Delta\tilde{y}$, then $\Delta\tilde{x}$, and finally $\Delta\tilde{z}$ from (3.25), (3.26), and (3.23), respectively. It consist of

- computing $r_1 := M_1^{-1}(\tilde{r}_x + C^T D_2 \tilde{r}_z)$ from (3.25) by performing a backsolve and a forward substitution: $L_1 L_1^T r_1 = \tilde{r}_x + C^T D_2 \tilde{r}_z$;
- computing $r_2 := A r_1 - \tilde{r}_y$;
- finding $\Delta\tilde{y}$ from (3.25) by performing a backsolve and a forward substitution: $L_2 L_2^T \Delta\tilde{y} = r_2$;
- computing the right-hand side $r_3 := -A^T \Delta\tilde{y} + \tilde{r}_x + C^T D_2 \tilde{r}_z$ from (3.26)
- finding $\Delta\tilde{x}$ from (3.26) by performing a backsolve and a forward substitution: $L_1 L_1^T \Delta\tilde{x} = r_3$;
- finding $\Delta\tilde{z}$ from (3.23).

Before describing the way CHOLMOD was integrated in our new linear algebra in OOQP, we give some of the main concepts related to the factorization of the (positive definite) matrices. The CHOLMOD factorization of a matrix is split in two parts. The first is the so-called symbolic analysis and consists of computations that typically depend only on the nonzero pattern, not the numerical values. The main duty of this phase is to find a permutation of the matrix so that the amount of fill-in in the factors is minimized (or at least significantly decreased). This is also known as finding the fill-reducing ordering. The symbolic analysis phase also includes the symbolic factorization, which consists of finding the explicit representation of the nonzero pattern of the factor(s). The second part of the CHOLMOD factorization process is the numerical factorization based on a Cholesky-based algorithm. An important observation is that the symbolic analysis is usually much more expensive than the numerical factorization.

There is a key aspect in using CHOLMOD in the context of interior-point methods. Since the numerical factorization is based on the Cholesky algorithm, no numerical pivoting is needed to maintain numerical stability. This implies that the permutation found by the symbolic analysis does not have to be recomputed when factorizing a matrix with different numerical values but the same sparsity pattern.

On the other hand, the matrices $M_1$ and $M_2$ from (3.26) and (3.25) have a special property that turns out to be crucial in our discussion. During the iterations of the interior-point method, only matrices $D_1$ and $D_2$ change. Since they are diagonal matrices (with positive diagonal entries), we obtain that the sparsity pattern of $M_1$ remains the same during the interior-point iterations. Consequently, $M_1^{-1}$ has the same pattern, which implies that the structure of $M_2$ remains the same. Our code incorporates the above observations. The symbolic analysis phase is done only once at the first iteration of the interior-point method. Any other subsequent factorization need is backed up by a fast CHOLMOD numerical factorization.

CHOLMOD 1.4 offers the possibility to choose between up to nine fill-reducing

ordering heuristics and two symbolic factorization methods. In our implementation, we let CHOLMOD decide on the best fill-reducing and symbolic factorization methods. Since both $M_1$ and $M_2$ have the special form $AA^T$, the usual choices of CHOLMOD were COLAMD for fill-reducing ordering and supernodal for symbolic factorization.

### 3.2. *TRON*

TRON is a trust region Newton method for bound constrained optimization problems. The algorithm uses a quadratic model function, projected searches during the subspace minimization phase and a preconditioned conjugate gradient method to determine the minor iterates. The limited memory preconditioner used is the incomplete Cholesky factorization of [38].

The Cauchy step at iteration $k$, $s_k^C$ is of the form $s_k(\alpha_k)$, where the function $s_k : \mathbb{R} \to \mathbb{R}^n$ is defined by

$$s_k(\alpha) = P\left[x_k - \alpha \nabla f(x_k)\right] - x_k.$$

Here $P$ is the projection onto the (bound constrained) feasible set, $x_k$ is the current iterate and $f$ the objective function. An iterative scheme that is guaranteed to terminate in a finite number of steps is used to compute the Cauchy point by generating a sequence $\left\{\alpha_k^{(l)}\right\}$ of trial values. The sequence can be either decreasing or increasing, based on the value of $\alpha_k^{(0)}$, where $\alpha_k^{(0)}$ is set to 1 in the (main-loop/major) first iteration and $\alpha_{k-1}$ otherwise.

Once the Cauchy point is obtained, a Newton step is sought subject to trust region constraints and with an active set choice determined by the one of the Cauchy point. If sufficient decrease is obtained compared to the one produced by the Cauchy point, the step is accepted. The algorithm is superlinearily convergent for nonlinear objective function.

### 3.3. *BLMVM*

BLMVM [14] is a projected gradient solver for nonlinear bound-constrained optimization problems. Like the unconstrained BFGS method, BLMVM creates a convex quadratic model function

$$m_k(d) = f(x_k) + \nabla f(x_k)^T d + \frac{1}{2} d^T B_k d,$$

where $f(x)$ is the objective function, $x_k$ is the current iterate, and the matrix $B_k$ is updated at each iteration using correction pairs $s_k$ and $y_k$. Unlike the unconstrained BFGS method, BLMVM defines the correction pairs $s_k$ and $y_k$ by

$$s_k = x_{k+1} - x_k, \quad y_k = T_\Omega \nabla f(x_{k+1}) - T_\Omega \nabla f(x_k).$$

Here $T_\Omega$ is the projection operator, with the $i$th component of $T_\Omega \nabla f(x)$ given by

$$\left(T_\Omega \nabla f(x)\right)_i = \begin{cases} \partial_i f(x) & \text{if } x_i \in (l_i, u_i) \\ \min\{\partial_i f(x), 0\} & \text{if } x_i = l_i \\ \max\{\partial_i f(x), 0\} & \text{if } x_i = u_i \end{cases}$$

where $\partial_i f(x)$ is the partial derivative of $f$ with respect to the $i$th variable $x_i$ and $\Omega = \{x \mid l \leq x \leq u\}$ is the bound constrained feasible set.

To reduce the cost of storing the inverse Hessian approximation, BLMVM uses the limited memory BFGS method (L-BFGS). The algorithm uses a projected line search to enforce the bounds on the variables.

### 3.4. *MOSEK*

The MOSEK Optimization Software (www.mosek.com) is a collection of tools for solution of large-scale optimization problems. MOSEK provides specialized solvers for linear programming, mixed integer programming, and many types of nonlinear and convex optimization problems, such as convex quadratic problems, conic quadratic problems, and quadratically constrained problems. In particular, for solving convex quadratic problems subject to linear constraints, MOSEK employs an homogeneous interior-point algorithm for monotone complementarity problems [1]. This homogeneous model is able to solve the problem without any regularity assumption on the existence of optimal, feasible or strictly interior feasible points. If the problem has a solution, the algorithm generates a sequence that approaches feasibility and optimality simultaneously. If the problem is infeasible, it generates a sequence that converges to a certificate proving infeasibility. The algorithm can start at any positive point (feasible or infeasible) and converges in no more than $O(\sqrt{n}\log(1/\epsilon)$ iterations, the best-known complexity for linear complementarity problems. In our experiments we used MOSEK 4.0 through the C optimizer API to solve the primal form (2.10).

## 4. Numerical Results

In the section we present details of applying the algorithms and solvers from Section 3.

### 4.1. *Environment and Solver Configuration*

We used RedHat Enterprise Linux 5 to run our experiments. The jobs were submitted to a SUN Grid Engine 6.0u8 running on 10 dual-processor computers each having between 2 GB and 4 GB of physical memory. All the processors in the grid are Pentium 4 at 3.06 MHz with 512KB L2 cache. Our simulation code uses one processor at the moment.

The simulation code was written in C and C++ and compiled by using GNU C and C++ compilers. The optimization solvers and their requirements were also compiled by using GNU tools. The code optimization flag was set to *-O3* for all compilers.

We used all software packages with their default stopping criteria. Changing these parameters may affect the conclusions but also make the results difficult to report. The difficulty of projected gradient methods, such as those BLMVM and TRON use, to obtain a solution for very stringent tolerance is well documented, and we do not investigate that here. Our goal is to provide a useful benchmark for engineering applications, which in many cases accept errors of the order of those in the stopping criteria of BLMVM and TRON. We thus believe that the fairest comparison of usefulness of these packages is for their default settings.

We used for both OOQP-MA27 and OOQP-CHOL the default stopping criteria, which consist of relative gap $\mu < 10^{-8}$ and relative norm of the residual less than

$10^{-8}$. The relative norm of the residual is the ratio between the norm of the residual and the absolute value of the largest magnitude element in the problem's data. We also run MOSEK with the default stopping parameters; namely, primal and dual feasibility tolerance and relative gap less than $10^{-8}$. TRON and BLMVM consider a problem solved when when the norm of the gradient falls under $10^{-4}$ and $10^{-3}$, respectively.

### 4.2. *Examples Generation*

In all of our experiments, in a unitless representation, the dimensions of the vat are 60 for the radius of the cylinder and 20 for the small radius of the truncated cone (see Figure 1). The height of the cylinder and the truncated cone are 80 and 40, respectively. The radius of each pebble is set to 1.

For the simulation experiments the pebbles are initially randomly arranged in horizontal planes. On each horizontal plane the pebbles are distributed in several inner circles. For the optimization experiments we also place pebbles on the bottom of the vat. The number of such bodies is approximately one-fourth the number of the suspended ones. Since we are interested in the situation when some of the falling balls are still in the air, while the others are interacting with the walls of the vat and with the pebbles from the bottom of the vat, the optimization problem is chosen after several seconds of simulation.

For any configuration, we solve the problem (2.13), which is set up as described in Section 2.1 for given $q^l, v^l$ and time step $h$. This produces the multipliers $\lambda^{(l+1)}$. We replace them in (2.9), to obtain $v^{(l+1)}$, the solution of (2.10). After this, the new position variables are obtained from $q^{(l+1)} = q^{(l)} + h\Gamma(q^{(l)})v^{(l+1)}$.

### 4.3. *Total Kinetic Energy Results*

It is well known that granular flow simulation is chaotic [41]. This means that the tiniest difference in position of the particles at a given time step is amplified exponentially in time. Therefore, comparing the outcome of the various solvers for individual particles is essentially hopeless beyond extremely small and few time steps. In order to compare the prediction of the various solvers, it may be more illuminating to use aggregate quantities. To that end, a meaningful quantity is the total kinetic energy.

The first plot of Figure 3 shows how the kinetic energy of a system consisting of 800 pebbles changes in time. The definition of the kinetic energy is $E(t) = \frac{1}{2}v(t)^T Mv(t)$. Its value was found by simulating the same configuration with the four solvers. In the second plot we represent the relative energy of found by OOQP-MA27, OOQP-CHOL, and MOSEK with respect to the energy found by BLMVM. The relative energy is $E_{rel} = \frac{|E_s - E_b|}{E_b}$, where $E_b$ and $E_s$ are the energies found by making use of BLMVM and one of the remaining solvers, respectively.

We note that the relative error in energy is insignificant, *in physical interpretation terms*, except after a large amount of simulation time. We also note, however, that this error occurs only at very small value of the kinetic energy (essentially, around the time the pebbles have stopped). They are due primarily to a small denominator; the corresponding absolute value is insignificant. In addition, as can be seen from Tables 4.1, 4.2, and 4.3 , some of the errors are due to the fact that BLMVM and TRON are iterative solvers that are stopped with larger error in both primal and dual than the interior point solvers.

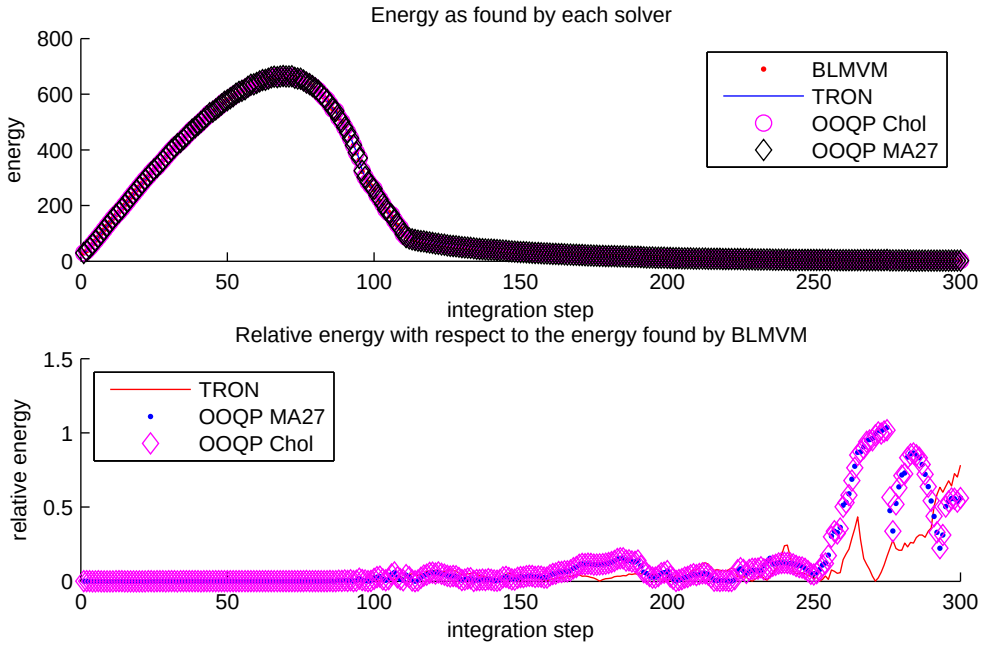



Figure 3. Energy dependence on time.

### 4.4. *Performance Results*

The tests involving MOSEK are performed on a Windows XP SP2 machine with 1.5GB memory running a P4 2.8 MHz processor with 512 KB L2 cache. On Windows, MOSEK was statically linked using Microsoft Visual Studio 7.1. We were not able to test MOSEK on Linux because only the Windows license was available to us. We are aware that the use of different hardwares and operating systems leads to different execution times. To have an idea about how big this difference is, we ran multiple simulations on both Windows and Linux computers. The simulations with OOQP-CHOL and BLMVM as the optimization solvers revealed that the execution is 30-35 percent slower on the Windows machine. One should keep in mind this difference when comparing execution times obtained by MOSEK on Windows with the execution times of the other four solvers on Linux.

We present two types of experiments comparing the solver performance. In the first experiment, the simulation test, we compare the solver performance for all the QPs encountered in the simulation, for different total numbers of pebbles. For such comparisons, QPs with the same number of dual variables are not necessarily the same. Therefore, different QPs with the same number of dual variables may be solved with different performance parameters, and the comparisons must be carried out only in terms of trends. This can be seen in the scatter plots of Figures 4 and 5.

A second experiment, the optimization test, progresses the simulation with one solver, OOQP-CHOL, up to a time where the QP to be solved is sufficiently large. At that point, the same QP is solved by all software packages, and the performance results are compared on the same problem.

#### *4.4.1. Simulation test*

Tables 4.1, 4.2, and 4.3 show the performance of each optimization solver in running simulations of 800, 1600, and 3200 pebbles, respectively. The second column indicates whether the primal (2.10) or dual (2.13) form was solved. The third col-

Table 4.1. Performance for 800 pebbles and $h = 0.05$.

| Solver | Pri/Dual | Primal Infeas | Int. steps | Total Time | Avg time |
|---|---|---|---|---|---|
| BLMVM | Dual | 1.482e-04 | 324 | 9738.880 | 30.058 |
| OOQP MA27 | Both | 0 | 370 | 59693.310 | 161.330 |
| OOQP Chol | Both | 0 | 371 | 9351.450 | 25.206 |
| TRON | Dual | 1.070e-02 | 487 | 282763.981 | 580.624 |
| Mosek* | Both | 0 | 407 | 1120797.148 | 2753.801 |

* Reported data is obtained on Windows.

Table 4.2. Performance for 1600 pebbles and $h = 0.05$

| Solver | Primal/Dual | Primal Infeas | Int. steps | Total Time | Avg time |
|---|---|---|---|---|---|
| BLMVM | Dual | 6.235e-05 | 394 | 73147.070 | 185.652 |
| OOQP MA27 | Both | 0 | 319 | 345773.404 | 1083.929 |
| OOQP Chol | Both | 0 | 310 | 38097.440 | 122.894 |

Table 4.3. Performance for 3200 pebbles and $h = 0.05$

| Solver | Primal/Dual | Primal Infeas | Int. steps | Total Time | Avg time | Kin energy |
|---|---|---|---|---|---|---|
| BLMVM | Dual | 4.946e-05 | 284 | 221411.620 | 779.618 | 2.3424 |
| OOQP Chol | Both | 0 | 296 | 175534.870 | 593.023 | 2.4757 |

umn, *Primal Infeas* lists the primal infeasibility at the last integration step. The number of integration steps needed to run the simulation is shown in the fourth column. The last two columns represent the total time in seconds needed for simulation and the average time in seconds per integration step. The simulations were stopped when the kinetic energy fell under a specific value: 0.2 for 800 pebbles, 0.8 for 1600, and 2.5 for 3200.

Both MOSEK and TRON crash while running the simulation involving 1600 pebbles. When solving the optimization problem from the first integration step MOSEK freezes in the preprocessing phase for several hours and then crashes. The memory usage before the abnormal termination is close to the maximum available. We believe that the lack of memory causes the failure of a memory allocation routine and consequently, MOSEK's crash. Although TRON was able to run the first several integration steps, as soon as the pebbles start to interact with the walls, and the size of the dual increases, it crashes. The source of the crash is a memory allocation failure in the FORTRAN 77 code. An important observation is that there is enough physical memory to satisfy the allocation request. Hence the failure is probably caused by FORTRAN 77 memory management routines. We present the memory use for BLMVM, OOQP, and TRON for this test in Figure 4.

TRON and MOSEK were not used for the experiment involving 3200 pebbles. The same simulation with OOQP MA27 was stopped because of the huge amount of time needed to solve the optimization problems (more than 20 times the time needed by OOQP-Chol for the same integration step).

### *4.4.2. Optimization test*

In the simulation test the solvers may solve different problems at each step since the system trajectories may be different due to accumulation of the numerical error. In this test, we compare the performance of all solvers for the same QP problem.

As described in Section 4.2, the optimization problems are chosen from a simulation of 1000 pebbles (800 staying on the bottom of the vat and 200 falling) after 3.3 seconds. The integration was done by using the timestep $h = 0.01$ for Table 4.4 and $h = 0.05$ for Table 4.5. Tables 4.4 and 4.5 list the solver name, the formulation solved, the number of unknowns in primal and dual, the number of iterations needed by each solver to solve the optimization problem, the average time per iteration, and the total time taken to solve the problem.

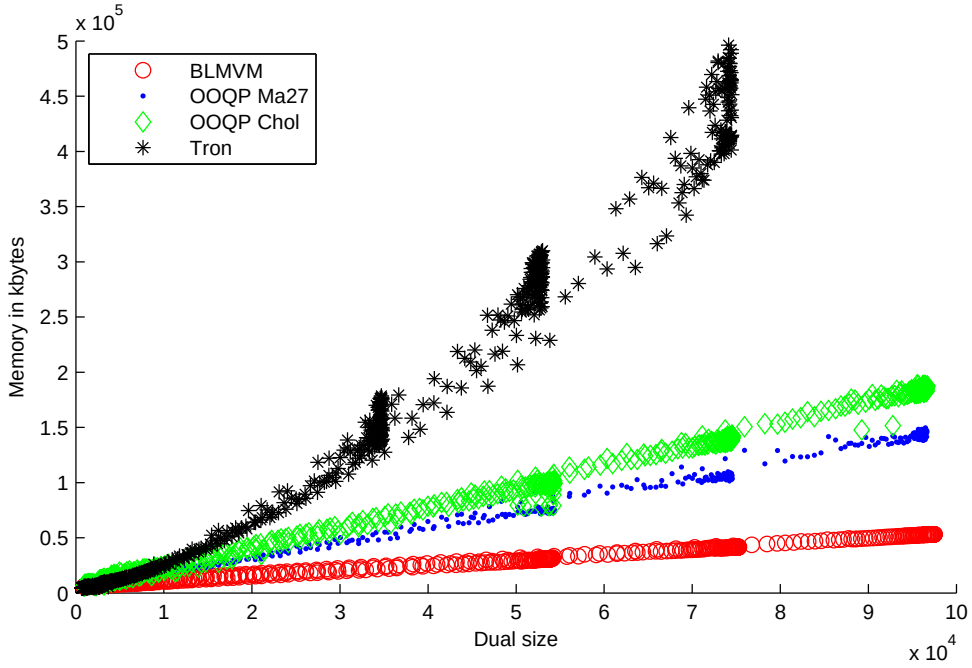



Figure 4. Memory performance.

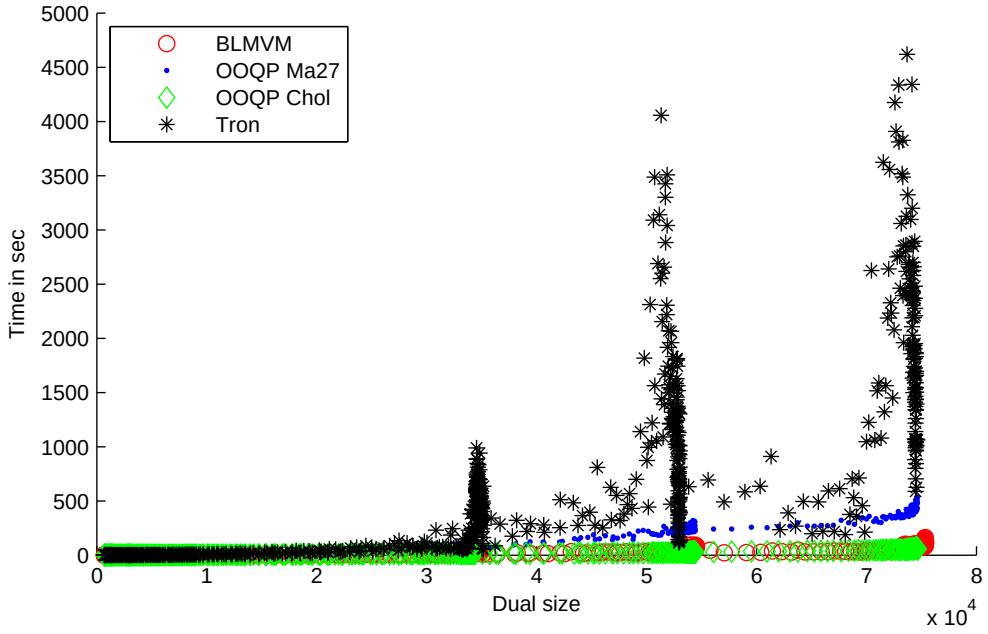



Figure 5. Execution time dependence on number of constraints.

Table 4.4. Optimization problem from the simulation of 1000 pebbles with $h = 0.01$

| Solver | Primal/Dual | Primal Size | Dual Size | No. Iter | Average Time | Total Time |
|---|---|---|---|---|---|---|
| BLMVM | Dual | 6000 | 62826 | 2501 | 0.127 | 318.016 |
| OOQP Ma27 | Primal/Dual | 6000 | 62826 | 33 | 21.120 | 696.984 |
| OOQP Chol | Prima/Dual | 6000 | 62826 | 33 | 3.115 | 102.812 |
| Mosek* | Primal/Dual | 6000 | 62826 | 24 | 465.362 | 11168.688 |
| TRON | Dual | 6000 | 62826 | | | CRASH |

* Reported data is obtained on Windows.

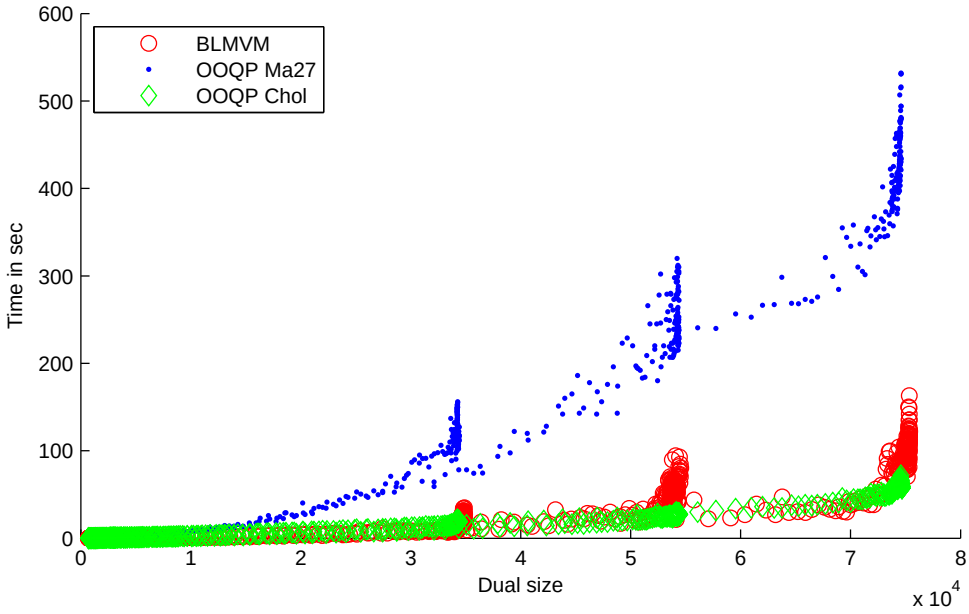



Figure 6. Execution time dependence on number of constraints.

Table 4.5. Optimization problem from the simulation of 1000 pebbles with $h = 0.05$

| Solver | Primal/Dual | Primal Size | Dual Size | No. Iter | Average Time | Total Time |
|---|---|---|---|---|---|---|
| BLMVM | Dual | 6000 | 62760 | 1729 | 0.128 | 221.640 |
| OOQP Ma27 | Primal/Dual | 6000 | 62760 | 31 | 21.351 | 661.881 |
| OOQP Chol | Prima/Dual | 6000 | 62760 | 31 | 3.016 | 93.500 |
| Mosek* | Primal/Dual | 6000 | 62760 | 24 | 456.534 | 10956.832 |
| TRON | Dual | 6000 | 62760 | | | CRASH |

* Reported data is obtained on Windows.

### 4.5. *Discussion of the Results*

We conclude that the ranking from most to least performing of the five solvers is OOQP-Chol (our linear algebra interface and implementation), BLMVM, OOQP-MA27, TRON, and MOSEK. This conclusion is sustained for both the simulation test, by results in Tables 4.1, 4.2 and 4.3 and in Figures 5 and 6, and the optimization test, by results in Tables 4.4 and 4.5.

The tabulated results show that this ordering holds on average, whereas the figures show that these results hold even when accounting for the spread of performance criteria for the same dual size. In addition, we also see from Figure 4 that both interior-point algorithms need more memory only by a factor of between 2 and 3 compared to BLMVM, which, as a limited-memory method, is quite memory-use conscious. From the memory results, it is also interesting to extrapolate what size of a problem will be held by a desktop. If the trends in Figure 4 hold, then a 4 GB architecture can hold a 150,000-pebble configuration, whereas a 32 GB architecture can hold a 600,000-pebble configuration.

We note that our OOQP-Chol implementation, using open source tools, is consistently a factor of 7 faster (and sometimes more than 20 times faster) compared to the OOQP-MA27 implementation. We also note that the time to solution performance of BLMVM and both OOQP implementation behaves fairly close to linear with the size of the problem. So both algorithms scale reasonably for this problem. In addition, our kinetic energy monitoring reveals that all solvers give comparable results.

Several caveats should accompany our conclusions. The first is that we do not re-

quire BLMVM to solve the problem to the same precision as for the interior-point solvers. Nonetheless, we believe that its results are useful, for reasons described in Subsection 4.1. The second is that, for license issues, we were not able to run MOSEK on the same architecture as the other solvers. Given our experience with the performance discount between Windows and Linux, we believe that the conclusions would not be change when running MOSEK on Linux, for reasons described in Section 4.4. We also note that the class of problems solved here, while of wide engineering interest, is limited insofar type of QPs encountered. For other QP types, it is conceivable that the performance ranking will change.

## 5. Conclusions and Future Work

We investigate the performance of four software packages for the resolution of quadratic programming problems with bound constraints that appear in the resolution of rigid multibody dynamics with contact and friction. These packages are TRON [38], BLMVM [14], MOSEK [1], and OOQP [29]. OOQP is investigated both with the default MA27 linear algebra and with our new implementation using Cholesky factorizations by means of the CHOLMOD package. We call the first instance OOQP-MA27 and the second OOQP-Chol.

We conclude that, for such problems, our OOQP-Chol implementation is the fastest of all the packages tested. It consistently uses only about three times more memory than BLMVM, while achieving far higher precision levels. Its behavior with the size of the problem is predictable, as can be seen from Figures 4, 5, and 6.

An important further research question is whether this performance holds for a parallel implementation. We note that a multithreaded version of CHOLMOD exists (see *http://www.cise.ufl.edu/research/sparse/cholmod/*) but does not currently exhibit good performance because of BLAS issues; these are expected to be fixed in the near future. The good parallel speedup of BLMVM, the closest competitor in terms of execution speed, is well documented [14].

Another important direction is to work directly with the disk constraint on the tangential force. This results in conic constraints on the contact force, which, in turn, leads to a quadratic program with conic constraints. We have recently shown that this formulation leads to a conic complementarity problem [11, 52]. We can solve this problem using a splitting scheme, of the Gauss-Seidel or Jacobi type [11]. At the moment, however, the set of codes available to us that support conic constraints includes only MOSEK, so we are not yet in the position to profile such codes for rigid multibody applications.

## Acknowledgments

Mihai Anitescu was supported by Contract No. DE-AC02-06CH11357 of the U.S. Department of Energy. The work of Florian Potra and Cosmin Petra was supported in part by the National Science Foundation under Grant CSE-0728878.

## References

[1] E. Andersen and Y. Ye, *On a homogeneous algorithm for the monotone complementarity problem*, Mathematical Programming, 84 (1999), pp. 375–399.

[2] M. Anitescu, *A fixed time-step approach for multibody dynamics with contact and friction*, Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on, 4 (2003).

[3] M. Anitescu, *Optimization-based simulation of nonsmooth dynamics*, Mathematical Programming, 105 (2006), pp. 113–143.
[4] M. Anitescu, J. Cremer, and F. Potra, *On the existence of solutions to complementarity formulations of contact problems with friction*, Complementarity and Variational Problems: State of the Art, (1997).
[5] M. Anitescu, J. F. Cremer, and F. A. Potra, *Formulating 3D contact dynamics problems*, Mechanics of Structures and Machines, 24(4) (1996), pp. 405–437.
[6] M. Anitescu and G. Hart, *A constraint-stabilized time-stepping approach for rigid multibody dynamics with joints, contact and friction.*, Int. J. Numer. Methods Eng., 60 (2004), pp. 2335–2371.
[7] M. Anitescu and G. D. Hart, *Solving nonconvex problems of multibody dynamics with joints, contact and small friction by sequential convex relaxation*, Mechanics Based Design of Machines and Structures, 31(3) (2003), pp. 335–356.
[8] ———, *A fixed-point iteration approach for multibody dynamics with contact and small friction*, Mathematical Programming, 101 (2004), pp. 3–32.
[9] M. Anitescu and F. A. Potra, *Formulating dynamic multi-rigid-body contact problems with friction as solvable linear complementarity problems*, Nonlinear Dynamics, 14 (1997), pp. 231–247.
[10] M. Anitescu, F. A. Potra, and D. Stewart, *Time-stepping for three-dimensional rigid-body dynamics*, Computer Methods in Applied Mechanics and Engineering, 177 (1999), pp. 183–197.
[11] M. Anitescu and A. Tasora, *An iterative approach for cone complementarity problems for nonsmooth dynamics*, Preprint ANL/MCS-P1413-0507, Argonne National Laboratory, Argonne, Illinois, 2007.
[12] D. Baraff, *Issues in computing contact forces for non-penetrating rigid bodies*, Algorithmica, 10 (1993), pp. 292–352.
[13] G. Bartels, T. Unger, D. Kadau, D. E. Wolf, and J. Kertsz, *The effect of contact torques on porosity of cohesive powders*, Granular Matter, 7 (2005), pp. 139–143.
[14] S. J. Benson and J. Moré, *A limited-memory variable-metric algorithm for bound-constrained minimization*, Tech. Rep. ANL/MCS-P909-0901, Mathematics and Computer Science Division, Argonne National Laboratory, 2001.
[15] A. Boeing and T. Bräunl, *Evaluation of real-time physics simulation systems*, Proceedings of the 5th international conference on Computer Graphics and Interactive Techniques in Australia and Southeast Asia, (2007), pp. 281–288.
[16] A. Chatterjee and A. Ruina, *A new algebraic rigid-body collision law based on impulse space considerations*, ASME, Transactions, Journal of Applied Mechanics, 65 (1998), pp. 939–951.
[17] T. A. Davis and W. Hager, *Dynamic supernodes in sparse cholesky update/downdate and triangular solves*, tech. rep., CISE Dept, Univ. of Florida, September 2006.
[18] T. A. Davis and W. W. Hager, *Modifying a sparse cholesky factorization*, SIAM Journal on Matrix Analysis and Applications, 20 (1999), pp. 606–627.
[19] T. A. Davis and W. W. Hager, *Row modifications of a sparse cholesky factorization*, SIAM Journal on Matrix Analysis and Applications, 26 (2005), pp. 621–639.
[20] S. Dirkse and M. Ferris, *The PATH solver: A non-monotone stabilization scheme for mixed complementarity problems*, Optimization Methods and Software, 5 (1995), pp. 123–156.
[21] B. R. Donald and D. K. Pai, *On the motion of compliantly connected rigid bodies in contact: a system for analyzing designs for assembly*, in Proceedings of the Conf. on Robotics and Automation, IEEE, 1990, pp. 1756–1762.
[22] I. S. Duff, *MA57—a code for the solution of sparse symmetric definite and indefinite systems*, ACM Trans. Math. Softw., 30 (2004), pp. 118–144.
[23] J. DUFF and J. REID, *The multifrontal solution of indefinite sparse symmetric linear equations*, ACM transactions on mathematical software, 9 (1983), pp. 302–325.
[24] K. Erleben, *Stable, Robust, and Versatile Multibody Dynamics Animation*, Ph.D. thesis, University of Copenhagen, Copenhagen, (2004).
[25] F.Camborde, C.Mariotti, and F.V.Donze, *Numerical study of rock and concrete behaviour by discrete element modeling*, Computers and Geotechnics, 27 (2000), pp. 225–247.
[26] R. Featherstone, *Robot Dynamics Algorithms*, Kluwer Academic Publishers, Boston, 1987.
[27] M. Ferris and T. Munson, *Interfaces to PATH 3.0: Design, implementation and usage*, Computational Optimization and Applications, 12 (1999), pp. 207–227.
[28] J. Gauvin, *A necessary and sufficient regularity condition to have bounded multipliers in nonconvex programming*, Mathematical Programming, 12 (1977), pp. 136–138.
[29] E. M. Gertz and S. J. Wright, *Object-oriented software for quadratic programming*, ACM Transactions on Mathematical Software, 29 (2003), pp. 58–81.
[30] H. D. Gougar, *Advanced core design and fuel management for pebble-bed reactors*, Ph.D in Nuclear Engineering, Department of Nuclear Engineering, Penn State University, 2004.
[31] N. I. M. Gould, J. A. Scott, and Y. Hu, *A numerical evaluation of sparse direct solvers for the solution of large sparse symmetric linear systems of equations*, ACM Trans. Math. Softw., 33 (2007), p. 10.
[32] D. Goulding, J.-P. Hansen, and S. Melchionna, *Size selectivity of narrow pores*, Physical Review Letters, 85 (2000), pp. 1132–1135.
[33] S. Hasegawa and M. Sato, *Real-time rigid body simulation for haptic interactions based on contact volume of polygonal objects*, Computer Graphics Forum, 23 (2004), pp. 529–538.
[34] E. J. Haug, *Computer-Aided Kinematics and Dynamics of Mechanical Systems*, Prentice-Hall, Englewood Cliffs, New Jersey, 1989.
[35] D. Helbing, I. Farkas, and T. Vicsek, *Simulating dynamical features of escape panic*, Nature, 407 (2000), pp. 487–490.
[36] W. Jorgensen, J. Chandrasekhar, J. Madura, R. Impey, and M. Klein, *Comparison of simple potential functions for simulating liquid water*, Journal of Chemical Physics, 79 (1983), pp. 926–935.
[37] D. Kadau, G. Bartels, L. Brendel, and D. Wolf, *Pore stabilization in cohesive granular systems*,

Phase Transitions: A Multinational Journal, 76 (2003), pp. 315–331.
[38] C. Lin and J. Moré, *Incomplete Cholesky Factorizations with Limited Memory*, SIAM Journal on Scientific Computing, 21 (1999), pp. 24–45.
[39] C.-J. Lin and J. J. Moré, *Newton's method for large bound-constrained optimization problems*, SIAM Journal on Optimization, 9 (1999), pp. 1100–1127.
[40] V. Milenkovic and H. Schmidl, *Optimization-based animation*, Proceedings of the 28th annual conference on Computer graphics and interactive techniques, (2001), pp. 37–46.
[41] R. Moeckel, *Chaotic dynamics near triple collision*, Archive for Rational Mechanics and Analysis, 107 (1989), pp. 37–69.
[42] J.-S. Pang, V. Kumar, and P. Song, *Convergence of time-stepping method for initial and boundary-value frictional compliant contact problems*, SIAM J. Numer. Anal., 43 (2005), pp. 2200–2226.
[43] J.-S. Pang and J. C. Trinkle, *Complementarity formulations and existence of solutions of dynamic multi-rigid-body contact problems with coulomb friction*, Math. Program., 73 (1996), pp. 199–226.
[44] F. A. Potra, M. Anitescu, B. Gavrea, and J. Trinkle, *A linearly implicit trapezoidal method for integrating stiff multibody dynamics with contact and friction*, International Journal for Numerical Methods in Engineering, 66(7) (2006), pp. 1079–1124.
[45] F. Radjai, M. J. J.-J. Moreau, and S. Roux, *Force distributions in dense two-dimensional granular systems*, Physical Review Letters, 77(2) (1996), pp. 274–277.
[46] C. Rycroft, M. Bazant, G. Grest, and J. Landry, *Dynamics of random packings in granular flow*, Physical Review E, 73 (2006), p. 51306.
[47] M. Saraniti, S. Aboud, and R. Eisenberg, *The simulation of ionic charge transport in biological ion channels: an introduction to numerical methods*, Reviews in Computational Chemistry, 22 (2006), pp. 229–293.
[48] P. Song, P. Kraus, V. Kumar, and P. Dupont, *Analysis of rigid-body dynamic models for simulation of systems with frictional contacts*, Journal of Applied Mechanics, 68(1) (2001), pp. 118–128.
[49] P. Song, J.-S. Pang, and V. Kumar, *A semi-implicit time-stepping model for frictional compliant contact problems*, International Journal of Numerical Methods in Engineering, 60 (2004), pp. 267–279.
[50] D. E. Stewart, *Rigid-body dynamics with friction and impact*, SIAM Review, 42(1) (2000), pp. 3–39.
[51] D. E. Stewart and J. Trinkle, *An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and coulomb friction*, International J. Numer. Methods Engineering, 39 (1996), pp. 281–287.
[52] A. Tasora and M. Anitescu, *ECCOMAS Thematic Conference in Multibody Dynamics*, Springer Verlag, Berlin, 2008, ch. A Fast NCP Solver for Large Rigid-Body Problems with Contacts, Friction, and Joints. To appear.
[53] A. Tasora, E. Manconi, and M. Silvestri, *Un nuovo metodo del simplesso per il problema di complementarit lineare mista in sistemi multibody con vincoli unilateri*, in Proceedings of AIMETA 05, Firenze, Italy, 2005.
[54] J. Trinkle, J.-S. Pang, S. Sudarsky, and G. Lo, *On dynamic multi-rigid-body contact problems with coulomb friction*, Zeithschrift fur Angewandte Mathematik und Mechanik, 77 (1997), pp. 267–279.
[55] A. Wallqvist and R. Mountain, *Molecular models of water: Derivation and description*, in Reviews in Computational Chemistry, vol. 13, John Wiley and Sons, 1999, pp. 183–247.
[56] W. W. H. Y. Chen, T. A. Davis and S. Rajamanicka, *Algorithm 8xx: CHOLMOD, supernodal sparse cholesky factorization and update/downdate*, tech. rep., CISE Dept, Univ. of Florida, September 2006.