Proceedings of the ASME 2013 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference
IDETC/CIE 2013
August 4-7, 2013, Portland Oregon, USA

DETC2013-13634

KRYLOV SUBSPACE METHODS FOR RIGID BODIES WITH COMPLIANT CONTACT AND COHESION

Hammad Mazhar
Department of Mechanical Engineering
University of Wisconsin Madison
Madison, WI 53706
Email: hmazhar@wisc.edu

ABSTRACT

Using particle based methods to simulate the behavior of compliant material is a complex task. When investigating the behavior of compliant terrain with hundreds of thousands of bodies in contact, millions of unknowns need to be determined. The size of problems solvable using traditional methods such as Jacobi or Gauss Seidel are severely limited due to the poor rate of convergence. This rate of convergence is typical when the equations of motion are posed as a differential variational inequality (DVI) problem that captures contact events between rigid bodies. The methods used for this framework rely on iterative krylov subspace methods such as Conjugate Gradient and Minimum Residual, which show good convergence for large problems. However, this class of iterative algorithms is not generally suitable for solving DVI problems for dynamics simulation. This document will show that these methods, while not specifically designed to solve rigid body dynamics problems, are very capable of doing so and converge very quicky.

1 INTRODUCTION

Increasingly engineers are using simulation techniques to augment and reduce time consuming and in some cases expensive experimental work. The goal of this project is to look into new and existing methods to try to understand how simulation techniques traditionally not used in granular dynamics can benefit large scale particle simulations. To this end a particles based dynamics engine was developed that is focused on terrain simulation, specifically the simulation of compliant, snow-like material. The engine supports spherical geometry of different radii and can simulate effects such as friction, compliance, and cohesion. This engine contains several iterative solvers that can be picked on the fly and used interchangeably, two time integration schemes are also available. A matrix free approach is used to reduce the memory footprint and simulate millions of objects without needing a prohibitive amount of memory. OpenMP is used throughout the engine to improve performance by utilizing multiple CPU cores.

Using parallel computing to reduce simulation time and/or simulate larger systems is a challenging task. Designing and implementing numerical methods capable on utilizing architectures with multiple compute cores remains an area of active research. Results reported in [5] show that popular methods for multi-body dynamics that are used in many commercial software packages are based on penalty or regularization approaches. These approaches run into difficulties when dealing with large problem sizes where hundreds of thousands of bodies and millions of contacts are present. Unlike penalty approaches the framework presented in this document is based on a differential variational inequality (DVI) approach which is capable of taking larger timesteps but has greater algorithmic complexity.

Copyright © 2013 by ASME

Deriving the Equations of Motion

Lets begin with the basic equations of motion (EOM):

$$\begin{aligned} m\ddot{x} &= F \\ J\ddot{\theta} &= n \end{aligned} \tag{1}$$

Here m is the mass of an object, $\ddot{x}$ is the acceleration, and F is the applied forces. For the second equation J is the moment of inertial, $\ddot{\theta}$ is the angular acceleration and n is the torque. From here we can write a simpler form where M represents the full mass matrix, $\ddot{q}$ is the linear and angular acceleration and Q is the applied forces and torques.

$$M\ddot{q} - Q = 0 \tag{2}$$

$$M = \begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & J \end{bmatrix}, \ddot{q} = \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{\theta} \end{bmatrix}, Q = \begin{bmatrix} F_x \\ F_y \\ n \end{bmatrix} \tag{3}$$

For a problem with constraints the vector Q contains both the applied forces Q^A and the constraint forces Q^C. The problem is that we do not know Q^C, the constraint forces, and therefore must get rid of them so that we can solve for them separately. To do this we have to first look at the variational form of the EOM.

$$\delta q^T[M\ddot{q} - (Q^A + Q^C)] = 0 \tag{4}$$

We already had that $M\ddot{q} - Q = 0$, if we multiply by a small displacement the equations should still hold true and this is where the variational form comes in. Now one key observation is that the net work produced by these small displacements for the constraint forces in the system is zero.

$$\delta q^T \cdot Q^C = 0 \tag{5}$$

This is true provided that the small displacement δq are consistent with the constraints. Say we have a consistent configuration q at time t^* that satisfies our constraints.

$$\Phi(q, t^*) = 0 \tag{6}$$

If we add a small displacement to q we get

$$\Phi(q + \delta q, t^*) = 0 \tag{7}$$

If we apply a Taylor series expansion, assuming small variations, we get:

$$\Phi(q + \delta q, t^*) \approx \Phi(q, t^*) + \Phi_q \delta q \Rightarrow \Phi_q \delta q = 0 \tag{8}$$

Because $\Phi(q, t^*) = 0$, $\Phi_q \delta q$ must also equal to zero. Now out goal was to separate Q to get just Q^A the applied forces. Because we state that $\delta q^T[M\ddot{q} - (Q^A + Q^C)] = 0$ condition hold true for a small displacement that is consistent with the set of constraints in our system we can get rid of Q^C and state:

$$\begin{aligned} \delta q^T[M\ddot{q} - Q^A] &= 0 \\ \Phi_q \delta q &= 0 \end{aligned} \tag{9}$$

In order to actually compute Q^C we must take a short detour and talk about Lagrange multipliers, specifically the Lagrange multiplier theorem (see appendix A).

Previously we had

$$\begin{aligned} \delta q^T[M\ddot{q} - Q^A] &= 0 \\ \Phi_q \delta q &= 0 \end{aligned} \tag{10}$$

Then using the Lagrange multiplier theorem if we set A, x and b to the following values:

$$\begin{aligned} x &\equiv \delta q \\ b &\equiv M\ddot{q} - Q^A \\ A &\equiv \Phi_q \end{aligned} \tag{11}$$

We get the following relationship:

$$A^T\hat{\lambda} + b = 0 \rightarrow \Phi_q^T\hat{\lambda} + (M\ddot{q} - Q^A) = 0 \tag{12}$$

Where $\Phi_q^T\hat{\lambda}$ is the constraint force, once the Lagrange multiplier has been computed this can be computed in a straightforward manner. Taking equation 12 we can write it in it's matrix form where γ is the constraint acceleration term.

$$\begin{bmatrix} M & \Phi_q^T \\ \Phi_q & 0 \end{bmatrix} \begin{bmatrix} \ddot{q} \\ \hat{\lambda} \end{bmatrix} = \begin{bmatrix} Q^A \\ \gamma \end{bmatrix} \tag{13}$$

This formulation is useful when dealing with kinematic, bilateral or holonomic constraints. When dealing with contact constraints however this formulation is unstable due to the large accelerations that can occur when two objects come into contact. A

Copyright © 2013 by ASME

better approach is to pose the constraint problem at the velocity level. This essentially take and force, applied or constraint, and makes it into an impulse. Taking equation 12 and multiplying through by out timestep h we get:

$$\Phi_q^T \hat{\lambda} h + (M\ddot{q} - Q^A)h = 0 \rightarrow \Phi_q^T \lambda + M\dot{q} - Q^A h = 0 \quad (14)$$

Where $\hat{\lambda}$ was used to compute the constraint force, λ is used to compute the constraint impulse. Similarly any applied forces became applied impulses. The constraint acceleration, γ becomes the constraint velocity ν. The updated matrix form becomes:

$$\begin{bmatrix} M & \Phi_q^T \\ \Phi_q & 0 \end{bmatrix} \begin{bmatrix} \dot{q} \\ \lambda \end{bmatrix} = \begin{bmatrix} Q^A h \\ \nu \end{bmatrix} \quad (15)$$

CONTACT CONSTRAINT PROBLEM

In order to pose our constrained problem with contacts we need to define Φ_q, the constraint Jacobian, and ν the constraint velocity.

Computing The Contact Jacobian

Computing Φ_q the constraint jacobian is straightforward for contacts. For each contact we have a contact normal n and an associated contact points s_A and s_B in the Global Reference Frame (GRF). First compute the representation of n in the Local Reference Frame (LRF), where A_A is the rotation matrix of body A. Note that for $\bar{n}_A$ the normal is $-n$.

$$\begin{aligned} \bar{n}_A &= A_A^T(-n) \\ \bar{n}_B &= A_B^T n \end{aligned} \quad (16)$$

Then compute the location of the contact points, s_A and s_B, in the LRF.

$$\begin{aligned} \bar{s}_A &= A_A^T(s_A - q_A) \\ \bar{s}_B &= A_B^T(s_B - q_B) \end{aligned} \quad (17)$$

Here q_A and q_B are the positions of both bodies in the GRF. Next we need to compute b the cross product of the contact points in the LRF, $\bar{s}_A$ and $\bar{s}_B$, with their associated transformed contact normals. b represents the moment arm of the contact force and forms the last three components of each half of the contact jacobian. An alternative to the cross product is to compute the skew symmetric form of $\bar{s}_A$ and $\bar{s}_B$ and multiply with $\bar{n}_A$ and $\bar{n}_B$.

$$\begin{aligned} b_A &= \tilde{\bar{s}}_A \bar{n}_A \\ b_B &= \tilde{\bar{s}}_B \bar{n}_B \end{aligned} \quad (18)$$

Where $\tilde{\bar{s}}$ is the skew-symmetric form of $\bar{s}$.

For contact i $\Phi_{q,i}$ is then:

$$\Phi_{q,i}^T = \begin{bmatrix} 0 \\ \vdots \\ -n_{i,A} \\ b_{i,A} \\ \vdots \\ n_{i,B} \\ b_{i,B} \\ 0 \end{bmatrix} \quad (19)$$

Contact Velocity ν

The velocity for a contact is the change in depth of penetration. If we define out gap function to be $\mathbf{g}$ then the contact velocity is simply $-\mathbf{g}/h$. The gap function represents the distance between two objects in contact, if it is negative then the two objects are in contact. Generally the gap is computed using collision detection.

The contact velocity term is actually the constraint stability term that is used to prevent constraint drift which can occur when two objects are resting on top of one another.

Complementarity And The Minimization Problem For Contact Constraints

Having derived the basic EOM for the dynamics problem we must pose it as a constrained optimization problem so that we can iteratively compute the λ values which will satisfy our constraints. This is done by writing the complementarity condition that represents the contact problem.

Taking equation 15 and replacing the contact velocity term nu with $-\mathbf{g}/h$ results in the following set of equations:

$$\begin{aligned} \Phi_q^T \lambda + M\dot{q} - Q^A h &= 0 \\ \Phi_q \dot{q} + \mathbf{g}/h &= 0 \end{aligned} \quad (20)$$

The second equation is used to write the complementarity condition for the constraint.

$$0 \leq \Phi_q \dot{q} + \mathbf{g}/h \perp \lambda \geq 0 \quad (21)$$

Copyright © 2013 by ASME

This condition states that if there is a contact and the gap between two bodies is less than or equal to 0, ($\Phi \leq 0$), then the contact impulse must necessarily be greater than 0, ($\lambda \geq 0$). Additionally this complementarity condition is the first order KKT condition for the following minimization problem. These complementarity conditions are what signifies that this is a DVI formulation of the rigid body dynamics problem which can be traced back to [4, 6, 8]

$$\min_{\lambda \geq 0} \frac{1}{2}\lambda^T N\lambda + b^T\lambda \tag{22}$$

This problem is a constrained optimization problem as the Lagrange multiplier, λ, must be greater than 0. This means that the contact impulse must be greater than zero and always push two objects apart. This formulation is known as a minimization problem with bound constraints. In order to compute the N and b values we must go back to our linear system in 15 and take it's shur compliment [1] to get:

$$\begin{gathered} N = \Phi_q M^{-1}\Phi_q^T \\ b = \mathbf{g}/h + \Phi_q M^{-1} Q^A h \\ N\lambda = b \end{gathered} \tag{23}$$

The complementarity condition from equation 21 can be re-written using N, b and λ [12].

$$0 \leq N\lambda - b \perp \lambda \geq 0 \tag{24}$$

Cohesion

Cohesion allows rigid bodies to stick to each other to a certain degree and is essentially a relaxation of the constrained optimization problem. Where $\lambda \geq 0$ becomes:. A more in depth discussion on the cohesion model is proided in [11].

$$\lambda \geq -compliance \tag{25}$$

Numerically this means that the contact impulse can be less than 0, causing two objects to stick together. The amount that two objects stick together can be controlled by the cohesion value. For very sticky behavior the constraint can be removed entirely making it an unconstrained optimization problem.

Compliance

Compliance controls the stiffness of a rigid body. This value is the inverse of stiffness, and a larger value means a softer object. Reducing the stiffness of a rigid body has two effects, first the contact becomes less rigid allowing more penetration to occur. Generally a value of 1e-5 results in stiff material with very little penetration. Secondly when compliance is reduced, i.e the material becomes stiffer, the system of equations becomes stiffer and thus more iterations are required to reach an acceptable solution. This modifications to the EOM are discussed in greater detail in [11]. After adding compliance the system of equations in 15 becomes:

$$\begin{bmatrix} M & \Phi_q^T \\ \Phi_q & E \end{bmatrix} \cdot \begin{bmatrix} \dot{q} \\ \lambda \end{bmatrix} = \begin{bmatrix} Q^A h \\ \frac{\mathbf{g}}{h+\alpha} \end{bmatrix} \tag{26}$$

Where E is the compliance matrix and is diagonal.

$$E = I \cdot \frac{k_i}{h(h+\alpha_i)} = \begin{bmatrix} \frac{k_0}{h(h+\alpha_0)} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \frac{k_i}{h(h+\alpha_i)} \end{bmatrix} \tag{27}$$

Here k_i is the compliance value (inverse of stiffness) and α_i is a scaling factor on the time-step h. α_i is usually between .1 and .2. Upon adding compliance equation 23 becomes:

$$\begin{gathered} N = (E - \Phi_q M^{-1}\Phi_q^T) \\ b = \frac{\Phi}{h+\alpha} + \Phi_q M^{-1} Q^A h \\ N\lambda = b \end{gathered} \tag{28}$$

Friction

The EOM along with the complementarity condition outlined above lead to a frictionless problem. This leads to more fluid-like behavior as the energy loss due to contact is purely through numerical damping and the fact that there is no restitution. Applying friction as an external force is straightforward and a model based on the tangential velocity of the two objects in contact can be used.

$$\begin{gathered} v_{BA} = v_{PB} - v_{PA} \\ v_n = (v_{BA} \cdot n) * n \\ v_t = |v_{BA} - v_n| \\ Q^A h = v_t \frac{\mu\lambda}{h} \end{gathered} \tag{29}$$

Here v_{PA} and v_{PB} are the velocities at the point of contact on each body. μ is the friction value and $\frac{\lambda}{h}$ is the normal force due to contact. This is a simplified model as it only models friction correctly during slip, doing so is straightforward however.

Copyright © 2013 by ASME

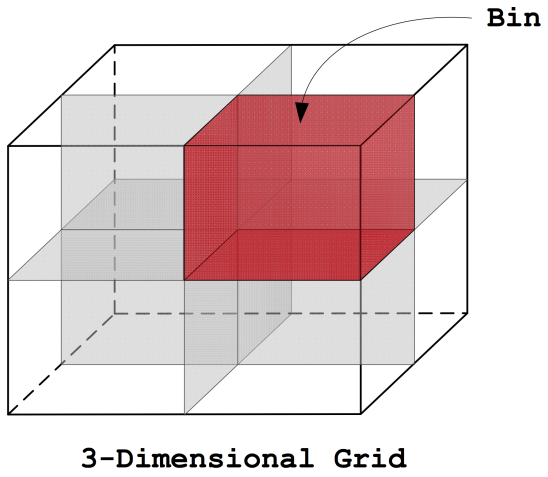

FIGURE 1. An example of a binned 3D space

Collision Detection

The collision detection algorithm used [2,7] is separated into two phases, broad-phase and narrow-phase. The broad-phase algorithm is used to compute whether two bodies might be in contact at a given time. The purpose of this algorithm is not to find actual contact information, but rather to determine if a contact could potentially occur based on the AABBs of the bodies involved. Once AABB-AABB intersections have been determined the narrow-phase algorithm performs a sphere-sphere contact check to see if an actual contact has occurred.

An Axis Aligned Bounding Box (AABB) is a special case of a bounding box that is always aligned to the global reference frame, simplifying collision detection as the bounding box cannot rotate. Because of this, the volume enclosed by the bounding box will always be equal to or greater than the volume of the shape it encloses. AABB generation is simple and can be easily paralellized on a per object basis. An AABB is computed by determining the maximum and minimum point of a shape. For example, an AABB for a sphere can be computed using its center and radius, with the minimum and maximum points being the radius subtracted and added to the center respectively.

A high-level overview of the collision detection algorithm is as follows. The collision detection process starts by identifying the intersections between AABBs and bins, see 1 for a visual representation of a bin. The AABB-bin pairs are subsequently sorted by bin id. Next, each bins starting index is determined so that the bins AABBs can be traversed sequentially. All AABBs touching a bin are subsequently checked against each other for collisions.

NUMERICAL SOLVERS

Due to the nature of the problem N is symmetric and positive semi-definite. The semi-definiteness is due to contact and in general constraint redundancy that can occur when multiple contacts act upon a body. The outcome of this is that there is no unique solution, several different solutions might lead to similar outcomes in terms of λ the constraint impulses.

Traditionally Jacobi or Gauss Seidel type solvers are used to solve the minimization problem. These solvers tend to converge very slowly as the problem size increases requiring thousands of iterations to converge to a meaningful solution. For this framework several Krylov-Subspace solvers were implemented as they tend to converge faster. Generally these solvers are not used for rigid body dynamics simulations as there is no unique solution and the fact that they can only solve Linear Complementarity Problems (LCPs), for frictionless material, compared to the more complex Cone Complementarity Problem (CCP) which can be used to implement a true friction cone rather than implementing friction as an external force.

Matrix Free Implementation

The computation of Nx where x is a vector is an extremely memory and compute intensive task. N is formed via equation 28. The computation of N is split into two stages. First $N_{temp} = \Phi_q M^{-1}$ is computed and stored in temporary variables for each object. Then $(E - N_{temp}\Phi_q^T)x$ is computed. Note that the second stage can be performed in parallel for each contact constraint. The first stage can also be parallelized but the process is not trivial as a reduction would be needed.

Implemented Solvers

The following solvers are available in the code. Each has the same input and output parameters allowing any solver to be switched in for a different one at any time.

Gradient Descent (GD)
Steepest Descent (SD)
Conjugate Gradient (CG)
Conjugate Gradient Squared (GS)
MinRes (Minres)
Quasi MinRes (QMR)
BiConjugate Gradient (BiCG)
BiConjugate Gradient with Stabilization (BiCGStab)

These methods will be compared in section 2 for a test case.

INTEGRATION SCHEMES

Semi-Implicit Euler

Semi-implicit Euler is the simpler of the two integrators implemented. It is Semi-implicit as only the velocity term is implicit, the position term is determined from velocity. This integrator is fairly stable but in cases with settled particles, small

Copyright © 2013 by ASME

amounts of jitter can occur.

$$\begin{aligned} v_{new} &= v_{old} + \Phi_q M^{-1} \lambda \\ x_{new} &= x_{old} + v_{new} h \end{aligned} \tag{30}$$

Backward Differentiation Formula (BDF)

The second order BDF formulation uses the values for velocity and position from the current and previous time-steps to determine the new velocity and position. The method is not fully implicit as position is computed via velocity. However, the method is more stable and generally fewer iterations are needed to converge as less jitter is present.

$$\begin{aligned} v_{new} &= \tfrac{4}{3} v_{current} - \tfrac{1}{3} v_{old} + \tfrac{2}{3} \Phi_q M^{-1} \lambda \\ x_{new} &= \tfrac{4}{3} x_{current} - \tfrac{1}{3} x_{old} + \tfrac{2}{3} v_{new} h \end{aligned} \tag{31}$$

Note that the time-step is scaled by $2/3$, and generally smaller time-steps are more stable, meaning that you get extra stability due to the artificially smaller time-step. Second order BDF is the default integration scheme used for this dynamics engine.

PARALLELIZATION

For problem sizes smaller than 10k objects single threaded code generally performs better than multi-threaded as there is some overhead associated with thread creating. With larger simulations there is benefit from using more than one thread.

OpenMP

OpenMP [10] is a library used for simplifying the task of parallelizing code by allowing the programmer to specify sections of code that can be parallelized. Loops that have no race conditions can be specified to use several threads in parallel to process. OpenMP is primarily used to parallelize loops that are either body or contact parallel in the code.

Thrust

Thrust [3] is a C++ template-based library based on the C++ Standard Template Library (STL). Using Thrust, parallel applications can be developed on the CPU or GPU with minimal effort, which result in high performance through a high-level API. The Thrust Library provides a collection of data parallel primitive algorithms such as scan, sort and reduce, among others. These simple operations can be combined together to implement complex algorithms with minimal source code. Because of its high-level API, Thrust can freely select the most optimal parameters for the GPU being used so that maximum performance is achieved. Algorithms are designed to be robust and are extremely useful for prototyping purposes.

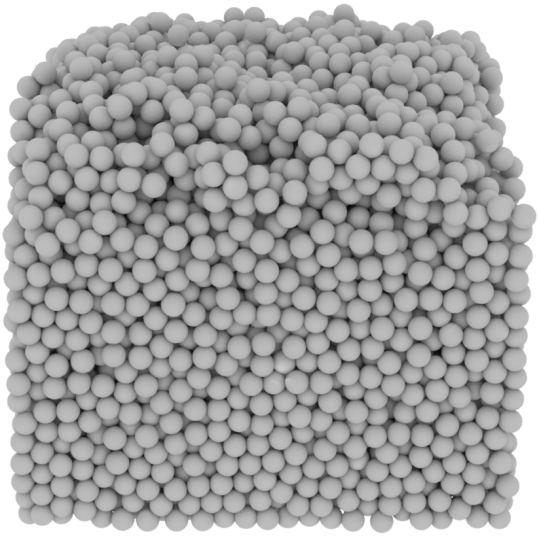

FIGURE 2. Image of settled rigid body configuration used for convergence analysis

Thrust is heavily used in the collision detection code to perform sort and scan operations. Thrust can automatically switch from OpenMP mode to CUDA [9] allowing for code to be ported between the CPU and GPU. Currently only the CPU version of thrust is used.

2 RESULTS

Convergence Analysis

In order to analyze the convergence characteristics of the solvers implemented in section 1 a settled configuration of rigid bodies was generated. This configuration was settled with full cohesion, meaning that the $\lambda \geq 0$ condition was neglected. This was done so that the λ values computed from the solvers were not modified, resulting in a more fair comparison. The settled configuration consisted of 10k particles with a compliance value of $1e{-}5$ and a solver tolerance of $1e{-}6$, note that the termination criteria of each solver is slightly different, in general the residual, or change in residual was used to terminate the algorithm. The results from this convergence analysis are presented in figure 3.

Results showed that the solver requiring the least number of iterations to converge is the BiConjugate Gradient method with Stabilization. Note that as the matrix N from section 1 is hermitian, the BiConjugate Gradient method is the same as the Conjugate Gradient method. The second fastest solution method was the Conjugate Gradient Squared method. Other noteworthy characteristics are that the convergence of the Minimum Resudual algorithm stalls after 50-60 iterations. Additionally as expected Gradient Descent was the worst solution method; Steepest Descent, by comparison converged much faster.

In practice the fastest numerical algorithm is Conjugate Gra-

Copyright © 2013 by ASME

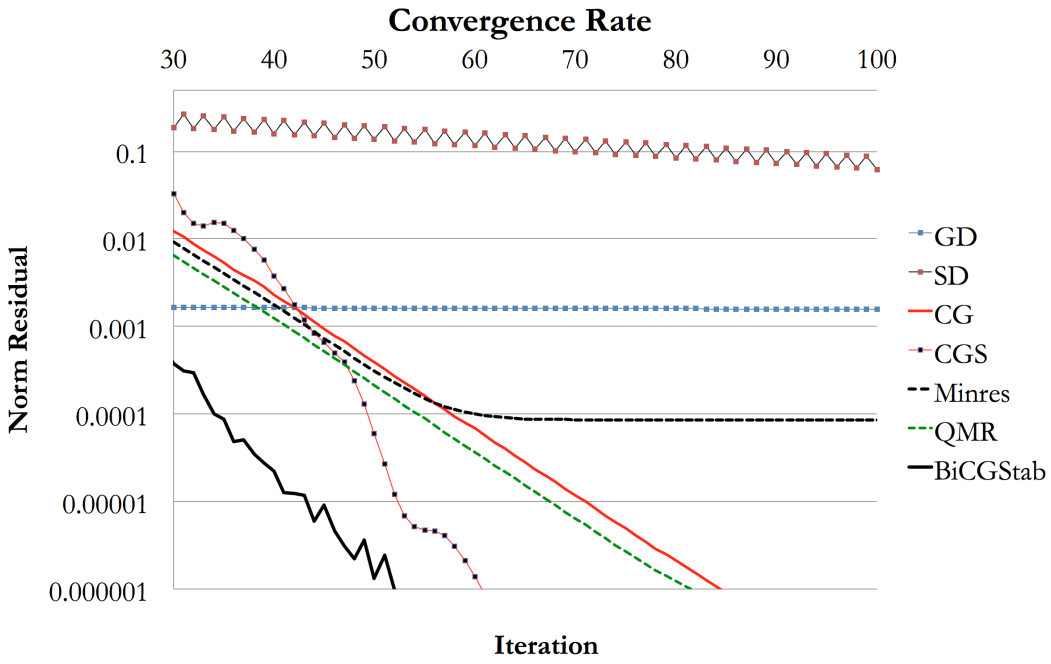

FIGURE 3. Comparison of different numerical solvers

dient as it requires only one matrix vector multiply. This sparse matrix vector multiplication (SPMV) is the bottleneck in the code and therefore solvers with fewer SPMV operations are much faster. In order for the CGS methods to be used it would need to converge in half the iterations as it has 2 SPMV operations.

Example Problems

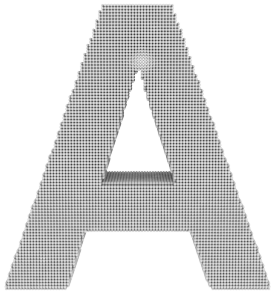

FIGURE 4. Initial configuration of particles. The data was generated using Houdini

Letter Drop The letter drop simulation is a basic scenario to show how compliance and cohesion affect the behavior of rigid bodies under the influence of gravity and contact. This example shows how the flow of particles can be controlled using cohesion. In the first simulation the particles flow freely and fill the bottom surface of the container. In the second simulation the combination of friction and cohesion act together to damp a lot of the energy that was present in the system. The initial configuration is shown in figure 4 and the final configurations are shown in 5.

Sticky Particles This simulation shows how rigid particles behave in interesting ways when cohesion, friction and compliance are added to the dynamics formulation. For this simulation a stream of particles falls onto a large static sphere. The entire simulation space is enclosed in a box and patterns in how the particles flow around the sphere can be seen. Due to the strong cohesive forces, the particles stick to the walls of the spheres in the same way that a fluid would. A fluid, due to it's viscosity and surface tension, sticks together rather than spread out. The same is true for this case where due to the cohesion and compliance of the bodies, particles clump together to form ribbons.

Walking in Snow These simulations demonstrate an actor interacting with a bed of rigid bodies that are compliant and have cohesion. Figure 7 shows the final frame of an actor walking through a block of snow with 200k particles. In this case the actor is invisible so that the snow particles can be better seen. Figure 8 shows a larger simulation with 1 million particles, where the snow is lower in comparison to the actor. In the first simulation it was ankle deep, in the second it is only to the feet of the actor. Some interesting characteristics to note are the bulldozing

Copyright © 2013 by ASME

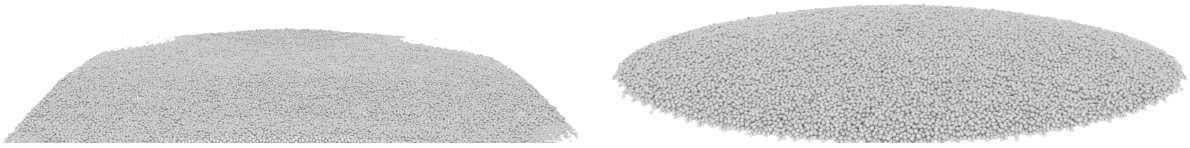

FIGURE 5. Final configuration of particles, image on left is without friction or cohesion and image on right is with friction and cohesion.

FIGURE 6. Sticky particles simulation

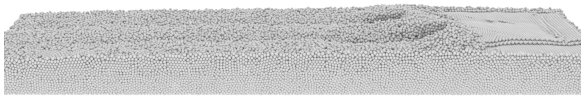

FIGURE 7. Result of actor walking in snow, simulation contains 200k particles with friction, and cohesion.

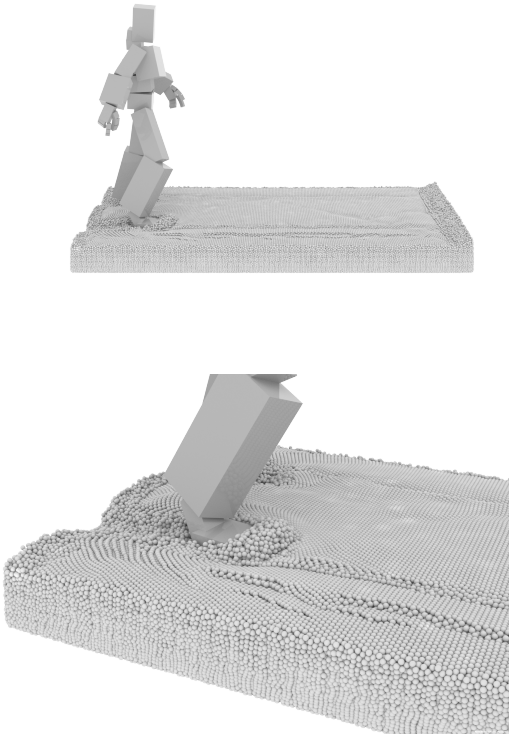

FIGURE 8. An actor interacting with 1 million snow particles, with friction and cohesion.

affects that are captured very well using this approach. Additionally as the actor walks through the snow cracks and ridges in the snow ahead of the actor begin to form which is a very interesting side effect of packing the initial set of spheres.

CONCLUSION

This paper describes a rigid body particle dynamics engine that is capable of simulating compliant material with friction and cohesion. By utilizing numerical methods traditionally not used for rigid body dynamics, such as conjugate gradient, problems with millions of unknowns can be solved much quicker than traditional solvers based on Gauss Seidel and Jacobi. Section 1 showed that the two fastest solvers to converge at the BiConjugate Gradient with Stabilization and Conjugate Gradient Squared methods. However, due to the fact that the sparse matrix vector multiplication operation, described in section 1, is a major bottleneck, the standard Conjugate gradient methods come out to be the fastest. Additionally a time integration method, 2nd order BDF, is more stable than semi-implicit euler when dealing with stacked particles. For these cases jitter was reduced leading to a more stable configuration.

Future work consists of porting the framework to the GPU via CUDA and Thrust to improve performance, along with optimizing the SPMV operation for better memory access on both the CPU and GPU. Additionally studying how to precondition the problem to reduce the overall number of iterations to converge will be very important for extremely large problems. In terms of modeling snow, varying the compliance value on a per particle basis might be one way to simulate compressibility and

Copyright © 2013 by ASME

will be investigated further. To facilitate larger simulations where an actor is interacting with the snow particles in a small region, and most of the particles are static, accelerating via a database or a simplified fluid solver will be investigated.

Appendix A: Lagrange Multiplier Theorem

Assume that we have a vector $b \in \mathbb{R}^{\kappa}$ and a matrix $A \in \mathbb{R}^{mxn}$ with $m < n$ such that for any vector $x \in \mathbb{R}^n$ for which the condition $Ax = 0$, the condition $x^T b = 0$ also holds. Essentially this means that any x that is perpendicular to the rows of A is automatically perpendicular to b. With this relationship it turns out that b is a linear combination of the rows of A and similarly the columns of A^T. In other words there is a "Lagrange Multiplier" λ that satisfies $b = -A^T\lambda$ and similarly $A^T\lambda + b = 0$.

REFERENCES

[1] J. Gallier. The schur complement and symmetric positive semidefinite (and definite) matrices. *December*, 44:1–12, 2010.

[2] Toby Heyn, Hammad Mazhar, Justin Madsen, Alessandro Tasora, and Dan Negrut. GPU-based parallel collision detection for granular flow dynamics. In *Proceedings of IDETC 09, San Diego*, 2009.

[3] J. Hoberock and N. Bell. Thrust: A Parallel Template Library, 2009. Available online at http://code.google.com/p/thrust/.

[4] P. Lotstedt. Mechanical systems of rigid bodies subject to unilateral constraints. *SIAM Journal of Applied Mathematics*, 42(2):281–296, 1982.

[5] J. Madsen, N. Pechdimaljian, and D. Negrut. Penalty versus complementarity-based frictional contact of rigid bodies: A CPU time comparison. Technical Report TR-2007-06, Simulation-Based Engineering Lab, University of Wisconsin, Madison, 2007.

[6] M. D. P. Monteiro Marques. *Differential Inclusions in Nonsmooth Mechanical Problems: Shocks and Dry Friction*, volume 9 of *Progress in Nonlinear Differential Equations and Their Applications*. Birkhäuser Verlag, Basel, 1993.

[7] H. Mazhar. *Parallel Multi-Body Dynamics on Graphics Processing Unit (GPU) Cards*. M.S. thesis, Department of Mechanical Engineering, University of Wisconsin–Madison, http://sbel.wisc.edu/documents/HammadMazharMSthesisFinal.pdf, 2012.

[8] Jean J. Moreau. Standard inelastic shocks and the dynamics of unilateral constraints. In G. Del Piero and F. Macieri, editors, *Unilateral Problems in Structural Analysis*, pages 173–221, New York, 1983. CISM Courses and Lectures no. 288, Springer–Verlag.

[9] NVIDIA Corporation. NVIDIA CUDA Developer Zone. Available online at https://developer.nvidia.com/cuda-downloads, 2012.

[10] OpenMP. Specification Standard 3.0. Available online at http://openmp.org/wp/, 2010.

[11] A. Tasora, M. Anitescu, S. Negrini, and D. Negrut. A compliant visco-plastic particle contact model based on differential variationalinequalities. *International Journal of Non-Linear Mechanics*, (0):–, 2013.

[12] Alessandro Tasora Toby Heyn, Mihai Anitescu and Dan Negrut. A comparison of several solvers for bound constraint quadratic programming within the context of frictionless multibody dynamics. Technical report, 2012.

Copyright © 2013 by ASME