Enabling High Performance Computational Dynamics in a Heterogeneous Hardware Ecosystem

Assistant Prof. Dan Negrut
Simulation-Based Engineering Lab
Department of Mechanical Engineering
University of Wisconsin – Madison

Toward High-Performance Computing Support for the Simulation and Planning of Robot Contact Tasks
June 27, 2011
Acknowledgements

- People:
  - Alessandro Tasora – University of Parma, Italy
  - Mihai Anitescu – Argonne National Lab
  - Lab Students:
    - Hammad Mazhar
    - Toby Heyn
    - Andrew Seidl
    - Arman Pazouki
    - Hamid Ansari
    - Justin Madsen
    - Naresh Khude
    - Makarand Datar
    - Dan Melanz
    - Markus Schmid

- Financial support
  - National Science Foundation, Young Investigator Career Award
  - FunctionBay, S. Korea
  - NVIDIA
  - Microsoft
  - Caterpillar
Talk Overview

- Overview of the engineering problems of interest

- Large-scale Multibody Dynamics
  - Problem formulation, solution method, and parallel implementation

- Overview of Heterogeneous Computing Template (HCT)

- Numerical Experiments

- Validation efforts

- Conclusions
Computational Multibody Dynamics

Simulation generated in ADAMS
Multi-Physics…
Fluid-Solid Interaction: Navier-Stokes + Newton-Euler.
Computational Dynamics
Rover Mobility on Granular Terrain

- Wheeled/tracked vehicle mobility on granular terrain
- Also interested in scooping and loading granular material
Frictional Contact Simulation
[Commercial Solution]

- Model Parameters:
  - Spheres: 60 mm diameter and mass 0.882 kg
  - Forces: smoothing with stiffness of 1E5, force exponent of 2.2, damping coefficient of 10.0, and a penetration depth of 0.1
  - Simulation length: 3 seconds

![Graph showing CPU time vs. Number of Spheres]

The graph shows the relationship between CPU time and the number of spheres, with the equation:

\[ y = 0.8385x^2 - 7.2607x + 16.154 \]

and

\[ R^2 = 0.9985 \]
Frictional Contact: Two Different Approaches Considered

- Discrete Element Method (DEM) - draws on a “smoothing” (penalty) approach
  - Lots of heuristics
  - Slow
  - General purpose
  - Used in ADAMS

- DVI-based (Differential Variational Inequalities)
  - A set of differential equations combined with inequality constraints
  - Fast (stable for significantly larger integration step-sizes)
  - Less general purpose
  - Used widely in computer games
The Modeling Component
Equations of Motion: Multibody Dynamics

\[ \dot{q} = T(q)v \]

**Generalized Positions**

**Kinematic Differential Equations**

**Force Balance Equations**

**Holonomic Kinematic Constraints**

**Contact Complementarity Conditions**

**Coulomb Friction Model**

**Velocity Transformation Matrix**

**Generalized Velocities**

**Frictional Contact Force**

\[ M(q)\ddot{v} = f(t, q, v) - g_q^T(q, t)\lambda + \sum_{i=1}^{N_c} (\gamma^i_nD^T_{n,i} + \gamma^i_uD^T_{u,i} + \gamma^i_wD^T_{w,i}) \]

**Reaction Force**

**Applied Force**

\[ g(q, t) = 0 \]

**Contact Impulse, for Contact “i”**

\[ 0 \leq \Phi^i(q, t) \perp \gamma^i_n \geq 0 \quad i = 1, 2, \ldots, N_c \]

**Total Number of Contacts**

**Gap Function, for Contact “i”**

\[ (\gamma_u^i, \gamma_w^i) = \arg \min_{\gamma_u^i, \gamma_w^i \geq 0} (\gamma_u^i v^T D_u^i + \gamma_w^i v^T D_w^i) \]

**Friction Impulse Components, for Contact “i”**

**Friction Dissipation Energy**
Traditional Discretization Scheme

\[ q^{(l+1)} = q^{(l)} + hL(q^{(l)})v^{(l+1)} \]

\[ M(v^{(l+1)} - v^l) = hf(t^{(l)}, q^{(l)}, v^{(l)}) + \sum_{i \in A(q^{(l)}, \delta)} \left( \gamma_i,n D_{i,n} + \gamma_i,u D_{i,u} + \gamma_i,w D_{i,w} \right) \]

For \( i \in A(q^{(l)}, \delta) \):

\[ 0 \leq \frac{1}{h} \Phi_i(q^{(l)}) + D_{i,n}^T v^{(l+1)} \perp \gamma_i,n \geq 0, \]

The minimization problem is:

\[ \gamma_i,u, \gamma_i,w = \arg\min_{\mu_i \gamma_i,n \geq \sqrt{\gamma_i,u^2 + \gamma_i,w^2}} v^T ( \gamma_i,u D_{i,u} + \gamma_i,w D_{i,w} ). \]

(Stewart & Trinkle, 1996)

- **Mass Mat.**
- **Applied Forces**
- **Reaction impulses**
- **Complementarity Condition**
- **Coulomb 3D friction model**
- **Stabilization term**
Relaxed Discretization Scheme Used

\[ q^{(l+1)} = q^{(l)} + hL(q^{(l)})v^{(l+1)} \]

\[ M(v^{(l+1)} - v^l) = hf(t^{(l)}, q^{(l)}, v^{(l)}) + \sum_{i \in A(q^{(l)}, \delta)} \left( \gamma_{i,n} D_{i,n} + \gamma_{i,u} D_{i,u} + \gamma_{i,w} D_{i,w} \right) \]

\[ \forall i \in A(q^{(l)}, \delta) : \quad 0 \leq \frac{1}{h} \Phi_i(q^{(l)}) + D_{i,n}^T v^{(l+1)} - \mu^i \sqrt{(v^T D_{i,u})^2 + v^T D_{i,w})^2} \perp \gamma_n^i \geq 0, \]

\[ (\gamma_{i,u}, \gamma_{i,w}) = \arg\min_{\mu_i \gamma_{i,n} \geq \sqrt{\gamma_{i,u}^2 + \gamma_{i,w}^2}} v^T (\gamma_{i,u} D_{i,u} + \gamma_{i,w} D_{i,w}) . \]

(Aniteşcu & Tasora, 2008)
The Cone Complementarity Problem (CCP)

- First order optimality conditions lead to Cone Complementarity Problem

- Introduce the convex hypercone...

\[
\Upsilon = \left( \bigoplus_{i \in \mathcal{A}(q^l, \epsilon)} \mathcal{F}C^i \right)
\]

\(\mathcal{F}C^i \in \mathbb{R}^3\) represents friction cone associated with \(i^{th}\) contact

... and its polar hypercone:

\[
\Upsilon^o = \left( \bigoplus_{i \in \mathcal{A}(q^l, \epsilon)} \mathcal{F}C^{i^o} \right)
\]

CCP assumes following form: Find \(\gamma\) such that

\[
\gamma \in \Upsilon \perp -(N\gamma + d) \in \Upsilon^o
\]
Putting Things in Perspective…

- Problem solved at each time step:
  (advancing simulation from $t_l$ to $t_{l+1}$)

$$\gamma \in \Upsilon \quad \perp \quad -(N\gamma + d) \in \Upsilon^o$$

$$v^{(l+1)} = M^{-1} \left( \tilde{k} + D\gamma \right)$$

$$q^{(l+1)} = q^{(l)} + hL(q^{(l)})v^{(l+1)} - (N\gamma + d) \in \Upsilon^o$$

- Three key points led to above algorithm:
  - Friction model posed as an optimization problem
  - Working with velocity and impulses rather than acceleration and forces
  - Contact complementarity expression altered to lead to CCP
Implementation

- Method outlined implemented using two loops
  - Outer loop – runs the time stepping
  - Inner loop – CCP Algorithm (solves CCP problem at each time step)
Granular Dynamics: How Parallel Computing is Leveraged

1. Parallel Collision Detection
2. (Body parallel) Force kernel
3. (Contact parallel) Contact preprocessing kernel
   4. (Contact parallel) CCP contact kernel
   5. (Constraint parallel) CCP constraint kernel
   6. (Reduction-slot parallel) Velocity reduction kernel
   7. (Body parallel) Body velocity update kernel
8. (Body parallel) Time integration kernel
Inner Loop (CCP Algorithm)

1. For each contact $i$, evaluate $\eta_i = 3 / \text{Trace}(D_i^T M^{-1} D_i)$.

2. If some initial guess $\gamma^*$ is available for multipliers, then set $\gamma^0 = \gamma^*$, otherwise $\gamma^0 = 0$.

3. Initialize velocities: $v^0 = \sum_i M^{-1} D_i \gamma^0_i + M^{-1} \bar{k}$.

4. For each contact $i$, compute changes in multipliers for contact constraints:

   \[
   \gamma_i^{r+1} = \lambda \Pi_{\gamma_i} \left( \gamma_i^r - \omega \eta_i \left( D_i^T v^r + b_i \right) \right) + (1 - \lambda) \gamma_i^r ;
   \]

   \[
   \Delta \gamma_i^{r+1} = \gamma_i^{r+1} - \gamma_i^r ;
   \]

   \[
   \Delta v_i = M^{-1} D_i \Delta \gamma_i^{r+1}.
   \]

5. Apply updates to the velocity vector:

   \[
   v^{r+1} = v^r + \sum_i \Delta v_i
   \]

6. $r := r + 1$. Repeat from 4 until convergence or $r > r_{max}$.
Large Scale Granular Dynamics

- Numerical solution can leverage parallel computing
CPU vs. GPU – Flop Rate (GFlop/Sec)

- Tesla 8-series
- Tesla 10-series
- Tesla 20-series

- Nehalem 3 GHz
- Westmere 3 GHz
- Single Precision
- Double Precision
CPU vs. GPU– Memory Bandwidth

[GB/sec]
Mixing 40,000 Spheres on the GPU
300K Spheres in Tank
[parallel on the GPU]
1.1 Million Rigid Spheres
[parallel on the GPU]
A Heterogeneous Computing Template for Computational Dynamics
Heterogeneous Cluster

Legend, Connection Type:

- Ethernet Connection
- Fast Infiniband Connection

Compute Node Architecture:

- CPU 0: Intel Xeon 5520
- CPU 1: Intel Xeon 5520
- RAM: 48 GB DDR3
- Hard Disk: 1TB
- Infiniband Card QDR

Gigabit Ethernet Switch

Network-Attached Storage

Second fastest cluster at University of Wisconsin-Madison
Computation Using Multiple CPUs

[DEM solution]
Computation Using Multiple CPUs

[DEM solution]
Computation Using Multiple CPUs

[DEM solution]
Use 1 GPU per MPI process

Each process uses its GPU to perform the collision detection task during each time step
  - As before, CPU takes care of communication between sub-domains
  - State data is copied to GPU, CD is performed, and collision data is copied back
  - GPU is used as an accelerator/co-processor
Demonstration

16,000 bodies
2 sub-domains

CPU:

CPU+GPU: 9.43 hrs
Heterogeneous Computing Template
Five Major Components

- Computational Dynamics requires
  - Domain decomposition
  - Proximity computation
  - Inter-domain data exchange
  - Numerical algorithm support
  - Post-processing (visualization)

- HCT represents the library support and associated API that capture this five component abstraction
Typical Simulation Results...

- LEFT: Infinity norm of the residual vs. iteration index in the CCP solution
  - Convergence rate (slope of curve) becomes smaller as the iteration index increases.

- RIGHT: Infinity norm of the CCP residual after $r_{\text{max}}$ iterations as function of granular material depth (number of spheres stacked on each other).
Searching for Better Methods

- Frictionless case (bound constraints in place)
  - Gauss-Jacobi (CE)
  - Projected conjugate gradient (ProjCG)
  - Gradient projected conjugate gradient (GPCG)
  - Gradient projected MINRES (GPMINRES)

- Friction case (cone constraints - ongoing)
  - Newton’s Method for large bound-constrained problems
    - Uses re-parameterization to handle friction cones (replace with bound constraints)
Numerical Experiments

- Test Problem: 40,000 bodies ⇒ 157,520 contacts
- Frictionless
Test Problem (MATLAB)

<table>
<thead>
<tr>
<th>Method</th>
<th>Iterations</th>
<th>Final Residual Norm</th>
<th>$\gamma_{\text{min}}$</th>
<th>$\gamma_{\text{max}}$</th>
<th>Time [sec]</th>
</tr>
</thead>
<tbody>
<tr>
<td>CE</td>
<td>1000</td>
<td>$6.11 \times 10^{-2}$</td>
<td>0.0</td>
<td>2.0598</td>
<td>1849.5</td>
</tr>
<tr>
<td>ProjCG</td>
<td>1002</td>
<td>$5.6344 \times 10^{-4}$</td>
<td>0.0</td>
<td>2.2286</td>
<td>1235.6</td>
</tr>
<tr>
<td>GPCG</td>
<td>1600</td>
<td>$1.0675 \times 10^{-4}$</td>
<td>0.0</td>
<td>2.6349</td>
<td>382.3644</td>
</tr>
<tr>
<td>GPMinres</td>
<td>1100</td>
<td>$9.5239 \times 10^{-5}$</td>
<td>0.0</td>
<td>2.3090</td>
<td>238.0744</td>
</tr>
<tr>
<td>PCG</td>
<td>1000</td>
<td>$2.4053 \times 10^{-4}$</td>
<td>-1.1116</td>
<td>2.5254</td>
<td>27.9686</td>
</tr>
<tr>
<td>GMRES</td>
<td>1000</td>
<td>$4.5315 \times 10^{-5}$</td>
<td>-1.1635</td>
<td>2.5227</td>
<td>736.3007</td>
</tr>
<tr>
<td>MINRES</td>
<td>1000</td>
<td>$1.6979 \times 10^{-5}$</td>
<td>-1.1316</td>
<td>2.5253</td>
<td>41.5790</td>
</tr>
</tbody>
</table>
Proximity Computation
GPU Collision Detection (CD)

- 30,000 feet perspective:
  - Carry out spatial partitioning of the volume occupied by the bodies
    - Place bodies in bins (cubes, for instance)
  - Follow up by brute force search for all bodies touching each bin
    - Embarrassingly parallel
Basic Idea: 
Search for Contacts in Different Bins in Parallel

- Example: 2D collision detection, bins are squares
Ellipsoid-Ellipsoid CD: Results

Time vs. Number of Contacts

Speedup - GPU vs. CPU
Speedup - GPU vs. CPU (Bullet library)  
[results reported are for spheres]  

GPU: NVIDIA Tesla C1060  
CPU: AMD Phenom II Black X4 940 (3.0 GHz)  

X Speedup  

Contacts (Millions)
Parallel Implementation: Number of Contacts vs. Detection Time [results reported are for spheres]
Multiple-GPU Collision Detection

Assembled Quad GPU Machine

Processor: AMD Phenom II X4 940 Black
Memory: 16GB DDR2
Graphics: 4x NVIDIA Tesla C1060
Power supply 1: 1000W
Power supply 2: 750W
SW/HW Setup

Main Data Set

Results

16 GB RAM

Open MP

CUDA

Quad Core AMD Microprocessor

Tesla C1060 4x4 GB Memory 4x30720 threads
Results – Contacts vs. Time

Quad Tesla C1060 Configuration

Contacts (Billions) vs. Time (Sec)
Conclusions

- Work aimed at enabling high-fidelity discrete models using a physics-based approach

- Approach draws on unique CPU + GPU parallel approach, which leverages a Heterogeneous Computing Template

- Accomplishments to date
  - Billion body parallel collision detection
  - Parallel solution of cone complementarity problem, about 12 million unknowns
  - Early validation results encouraging

- Aiming at billion bodies simulations
Ongoing/Future Work

- Massively parallel linear algebra for solution of CCP problem
- More general collision detection code
- Multiphysics:
  - Fluid-solid interaction
  - Electrostatics
Thank You.
Validation.

Simulation is doomed to succeed.
(Rod Brooks, roboticist)
- Validation at “microscale” – University of Wisconsin-Madison
  - Work in progress

- Validation at “macroscale” – University of Parma, Italy
Flat Hopper Tests

Video recording from a test (a case that starts from high crystallization)
Flat Hopper Tests

3D rendering from a simulation (4x slower than real-time)
Flat Hopper Tests

- Comparison experimental - simulated
Validation at Microscale

- Sand flow rate measurements
- Approx. 40K bodies
- Glass beads
- Diameter: 100-500 microns
Experimental Setup

- CPU connection
- Disruptor beads
- Nanopositioner controller
- Load cell
- Translational stage
- Nanopositioner
Flow Measurement, 500 micron Spheres
Flow Simulation, 500 micron Spheres
Flow Measurement Results, 3mm Gap Size
Flow Measurement Results, 2.5mm Gap Size

![Graph showing weight [N] over time [sec] for different runs (1 to 8) with a 2.5mm gap size.](image)
Flow Measurement Results, 2mm Gap Size

![Graph showing weight vs. time for different trials with a 2mm gap size.](image-url)
Flow Measurement Results, 1.5mm Gap Size
Validation Experiment: Repose Angle

- Experiment
- Simulation

\[ \phi = 19.5^\circ \quad \text{for} \quad \mu = 0.39 \]
Validation Experiment Flow and Stagnation
Validation, Flow and Stagnation
Validation, Flow and Stagnation

Gapsize 2.0mm

Weight in N

Time in s

- 1
- 2
- 3
- 4
- 5
- 6
- \(\mu = 0.15\)
Spherical Decomposition

- Represent complex geometry as a union of spheres
  - Fast parallel collision detection on GPU
  - Allows non-convex geometry
  - NOTE: Used only for CD, the dynamics is performed using original geometry
Examples...

- Chain model
  - 10 links
  - 7,797 spheres per link

- Plow model
  - 31,791 spheres in plow blade model
  - 15,000 spheres representing terrain
Numerical Results: Pebble Bed Nuclear Reactor

- Two types of tests were run

- On the GPU
  - CD: NVIDIA 8800 GT
  - CCP: NVIDIA Tesla C870

- On the CPU
  - Single threaded
  - Quad Core Intel Xeon E5430
  - 2.66 GHz

- The reactor contains spheres which flow out the bottom the nozzle and are recycled to the top of the reactor

- Performed simulations with 16K, 32K, 64K, and 128K bodies
Results: Average Duration for Taking One Integration Step [Δt=0.01 s]

Pebble Bed Reactor Demo, GPU vs. CPU

- **GPU Average Total Time**
- **CPU average Total Time**
- Linear (GPU Average Total Time)
- Linear (CPU average Total Time)

Equations:
- **GPU**: \(y = 0.0007x - 6.3447\), \(R^2 = 0.996\)
- **CPU**: \(y = 6E-05x - 0.51\), \(R^2 = 0.9936\)
Algorithmic Challenges: Gauss-Jacobi Doesn’t Scale Well

- Convergence stalls for certain classes of problems
  - Very large problems

- Problems where I have many body stacked on each other
  - Inherent problem with Gauss-Jacobi, pertains the propagation of information

- Granular dynamics problem have an intrinsic degree of redundancy
Ellipsoid-Ellipsoid CD: Visualization
Example: Ellipsoid-Ellipsoid CD

\[ d = P_1 - P_2 = \left( \frac{1}{2\lambda_1} M_1 + \frac{1}{2\lambda_2} M_2 \right) c + (b_1 - b_2) \]

\[ \frac{\partial d}{\partial \alpha_i} = \frac{\partial P_1}{\partial \alpha_i} - \frac{\partial P_2}{\partial \alpha_i}, \quad \frac{\partial^2 d}{\partial \alpha_i \partial \alpha_j} = \frac{\partial^2 P_1}{\partial \alpha_i \partial \alpha_j} - \frac{\partial^2 P_2}{\partial \alpha_i \partial \alpha_j} \]

\[ \frac{\partial P}{\partial \alpha_i} = \left( \frac{1}{2\lambda} M - \frac{1}{8\lambda^3} M c^T M \right) \frac{\partial c}{\partial \alpha_i} \]

\[ \frac{\partial^2 P}{\partial \alpha_i \partial \alpha_j} = \left( -\frac{1}{8\lambda^3} M + \frac{3}{32\lambda^5} M c^T M \right) c^T M \frac{\partial c}{\partial \alpha_j} + \frac{\partial^2 c}{\partial \alpha_i \partial \alpha_j} \]

\[ \lambda^2 = \frac{1}{4} n^T M n \]

\[ \varepsilon : \frac{x^2}{r_1^2} + \frac{y^2}{r_2^2} + \frac{z^2}{r_3^2} = 1 \]

\[ \lambda : \text{Rotation Matrix} \]

\[ M = AR^2A^T \]

\[ R = \text{diag}(r_1, r_2, r_3) \]

\[ b : \text{Translation of ellipsoids center} \]

\[ d = P_1 - P_2 \]

\[ \min_{\alpha_1, \alpha_2} \|d(\alpha_1, \alpha_2)\|^2 \]
Going Back to Practical Problem of Interest

- Tracked Vehicle on Granular Terrain…
Spherical Decomposition Steps

1: Take shoe element

2: Cubit

3: Spherical Padding
Track Components

1,594,908 spheres per track
Granular Terrain Model

- Represent terrain as collection of discrete particles
- Match terrain surface profile
- Capture changing granularity with depth
Track Simulation 1

Parameters:
- Driving speed: 1.0 rad/sec
- Length: 12 seconds
- Time step: 0.005 sec
- Computation time: 18.5 hours
- Particle radius: 0.027273 m
- Terrain: 284,715 particles
Track Simulation 2

Parameters:

- Driving speed: 1.0 rad/sec
- Length: 10 seconds
- Time step: 0.005 sec
- Computation time: 17.8 hours
- Particle radius: .025±.0025 m
- Terrain: 467,100 particles
Results: Track ‘Footprint’
Results: Positions

Track Simulation 1