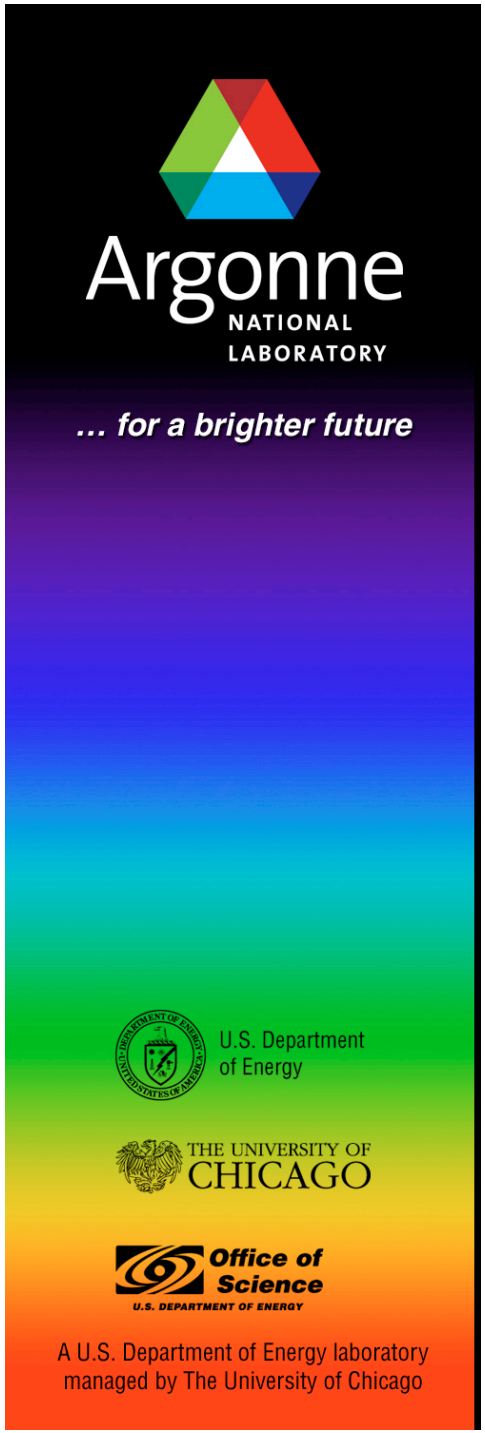


Iterative Methods for Optimization Problems in Rigid-body Simulation

Mihai ANITESCU ,

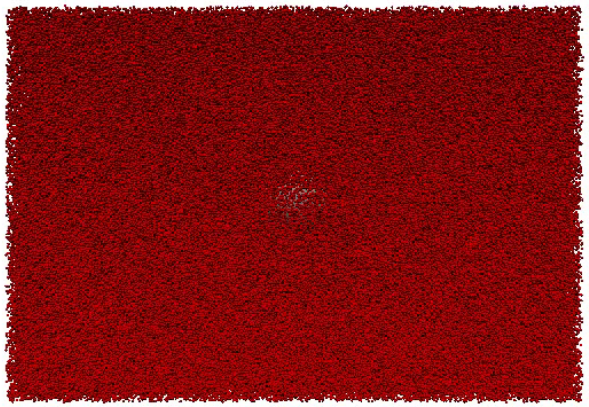
Argonne National Laboratory

SIAM PP 2010

With D. Negrut, A. Tasora, T. Heyn, A. Mazhar

Motivation: Granular Materials

- The motion of a large, dense set of rigid particles: grand challenge of physics.
- Despite centuries of study, and interest from DaVinci, Newton, there is no satisfactory continuum theory or hybrid theory.
- Difficulty: co-existing gas, liquid, and solid phases.
- I will use granular materials/ dynamics for “rigid multibody dynamics”



1 million particle simulation

Granular Materials: Perspectives

- Important? The second-most manipulated material in industry after water (Richard, Nature Materials 2005).
- Applications range from pharmaceutical, food, powders, petrochemical, nuclear, automotive, and semiconductor industries up to geological granular flows – some examples later.
- The absence of a continuum theory makes particle-by-particle computational approaches the only general first principles computing approach – we need HPC (1 cubic meter of sand has 1 trillion particles)

(Dry) Granular materials: equations

- Equations of motion: mixture of ordinary differential equations and variational inequalities/complementarity conditions.

Newton Equations

Non-Penetration Constraints

$$M\frac{dv}{dt} = \sum_{j=1,2,\ldots,p}\left(c_n^{(j)}n^{(j)} + \beta_1^{(j)}t_1^{(j)} + \beta_2^{(j)}t_2^{(j)}\right) + f_c(q,v) + k(t,q,v)$$

$$\frac{dq}{dt} = \Gamma(q)v$$

Generalized Velocities

$$c_n^{(j)} \geq 0 \quad \perp \quad \Phi^{(j)}(q) \geq 0, \qquad j = 1,2,\ldots,p$$

$$\left(\beta_1^{(j)}, \beta_2^{(j)}\right) = \operatorname{argmin}_{\mu^{(j)}c_n^{(j)} \geq \sqrt{\left(\beta_1^{(j)}+\beta_2^{(j)}\right)^2}} \left[\left(v^T t_1^{(j)}\right)\beta_1 + \left(v^T t_2^{(j)}\right)\beta_2\right]$$

Friction Model

Granular materials: abstraction: DVI

- Differential variational inequalities Mixture of differential equations and variational inequalities.

$$\begin{aligned} y' &= f(t, y(t), x(t)) \\ x(t) &\in SOL(K; F(t, y(t), \cdot)) \\ y(0) &= y_0 \end{aligned}$$

$$x \in SOL(K; F(t, y, \cdot) \Leftrightarrow (\tilde{x} - x)^T F(t, y, x) \geq 0, \forall \tilde{x} \in K$$

- Target Methodology (only hope for stability): time-stepping schemes.

$$\begin{aligned} y^{h,(i+1)} &= y^{h,i} + h\tilde{f}\left(\tilde{t}^{h,(i+1)}, \theta_1 y^{h,i} + (1-\theta_1) y^{h,(i+1)}, x^{h,(i+1)}\right) \\ x^{h,(i+1)} &\in SOL(K; \tilde{F}(\tilde{t}^{h,(i+1)}, \theta_2 y^{h,i} + (1-\theta_2) y^{h,(i+1)}, \cdot)) \\ y(0) &= y_0. \end{aligned}$$

Our Inquiry

- Can we derive iterative, and thus suitable for parallelism algorithms to simulate large-scale DVI, particularly granular dynamics, in a time-stable manner while being truthful to the physics of the respective applications?
- Stability => Time Stepping, as opposed to smoothing, DEM

Step 1 on the road. Time stepping scheme with fixed time step (no collision stop/restart)

- A measure differential inclusion solution can be obtained by time-stepping (Stewart, 1998, Anitescu, 2006): *PEC (approximated by LCP)

$$M(\boldsymbol{v}^{(l+1)} - \boldsymbol{v}^{l}) = \sum_{i \in \mathcal{A}(q^{(l)}, \epsilon)} \left(\gamma_n^i \boldsymbol{D}_n^i + \gamma_u^i \boldsymbol{D}_u^i + \gamma_v^i \boldsymbol{D}_v^i\right) +$$

$$+ \sum_{i \in \mathcal{G}_{\mathcal{B}}} \left(\gamma_b^i \nabla \Psi^i\right) + h \boldsymbol{f}_t(t^{(l)}, \boldsymbol{q}^{(l)}, \boldsymbol{v}^{(l)})$$

Speeds

Reaction impulses

Forces

Stabilization terms

$$0 = \frac{1}{h}\Psi^i(\boldsymbol{q}^{(l)}) + \nabla \Psi^{i^T} \boldsymbol{v}^{(l+1)} + \frac{\partial \Psi^i}{\partial t}, \quad i \in \mathcal{G}_{\mathcal{B}}$$

Bilateral constraint equations

$$0 \leq \frac{1}{h}\Phi^i(\boldsymbol{q}^{(l)}) + \nabla \Phi^{i^T} \boldsymbol{v}^{(l+1)}$$

Contact constraint equations

$$\perp \quad \gamma_n^i \geq 0, \ i \in \mathcal{A}(q^{(l)}, \epsilon)$$

COMPLEMENTARITY!

$$(\gamma_u^i, \gamma_v^i) = \operatorname{argmin}_{\mu^i \gamma_n^i \geq \sqrt{(\gamma_u^i)^2 + (\gamma_v^i)^2}} \quad i \in \mathcal{A}(q^{(l)}, \epsilon)$$

$$\left[\boldsymbol{v}^T(\gamma_u \boldsymbol{D}_u^i + \gamma_v \boldsymbol{D}_v^i)\right]$$

Coulomb 3D friction model

$$\boldsymbol{q}^{(l+1)} = \boldsymbol{q}^{(l)} + h\boldsymbol{v}^{(l+1)},$$

Pause: Efficiency

- This scheme allows fixed time steps for plastic collisions, major improvement in efficiency.
- Nevertheless, the PATH solver very useful and used in computer graphics (Lemke, exponential worst-case complexity), starts to take extremely long times past ~1000s of granules (Anitescu and Hart 04)… and we aim for 1 trillion.
- The main difficulty: the time-stepping subproblem is not convex.
- Question: How do I create an algorithm that has a prayer to scale?
- Answer: Relax the problem inspired by the physics.

Time Stepping -- Convex Relaxation– Step 2

- A modification (relaxation, to get convex QP with conic constraints):

$$M(\boldsymbol{v}^{(l+1)} - \boldsymbol{v}^l) = \sum_{i \in \mathcal{A}(q^{(l)}, \epsilon)} \left(\gamma_n^i \boldsymbol{D}_n^i + \gamma_u^i \boldsymbol{D}_u^i + \gamma_v^i \boldsymbol{D}_v^i\right) +$$

$$+ \sum_{i \in \mathcal{G}_\mathcal{B}} \left(\gamma_b^i \nabla \Psi^i\right) + h \boldsymbol{f}_t(t^{(l)}, \boldsymbol{q}^{(l)}, \boldsymbol{v}^{(l)}) \qquad (\text{}$$

$$0 = \frac{1}{h}\Psi^i(\boldsymbol{q}^{(l)}) + \nabla \Psi^{i^T} \boldsymbol{v}^{(l+1)} + \frac{\partial \Psi^i}{\partial t}, \quad i \in \mathcal{G}_\mathcal{B} \quad (\text{}$$

$$0 \leq \frac{1}{h}\Phi^i(\boldsymbol{q}^{(l)}) + \nabla \Phi^{i^T} \boldsymbol{v}^{(l+1)} \quad \boxed{-\mu^i \sqrt{(\boldsymbol{D}_u^{i,T}\boldsymbol{v})^2 + (\boldsymbol{D}_v^{i,T}\boldsymbol{v})^2}}$$

$$\perp \quad \gamma_n^i \geq 0, \; i \in \mathcal{A}(q^{(l)}, \epsilon)$$

$$(\gamma_u^i, \gamma_v^i) = \operatorname{argmin}_{\mu^i \gamma_n^i \geq \sqrt{(\gamma_u^i)^2 + (\gamma_v^i)^2}} \quad i \in \mathcal{A}(q^{(l)}, \epsilon)$$

$$[\boldsymbol{v}^T(\gamma_u \boldsymbol{D}_u^i + \gamma_v \boldsymbol{D}_v^i)] \qquad (\text{}$$

$$\boldsymbol{q}^{(l+1)} = \boldsymbol{q}^{(l)} + h\boldsymbol{v}^{(l+1)}, \qquad (\text{}$$

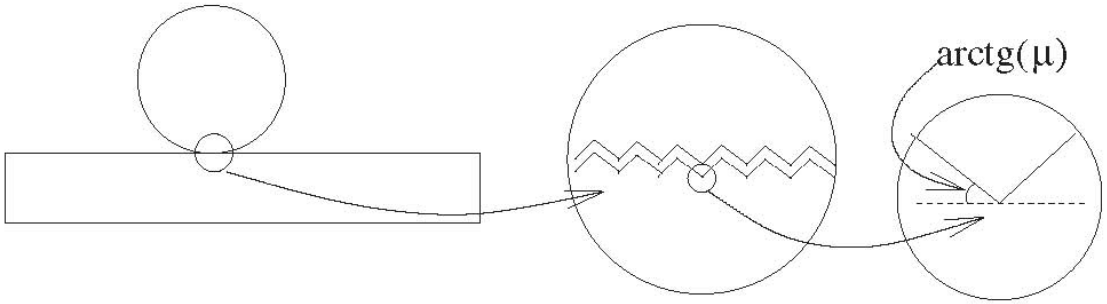
(For small m and/or small speeds, almost no one-step differences from the Coulomb theory)

But In any case, converges to same MDI as unrelaxed scheme.

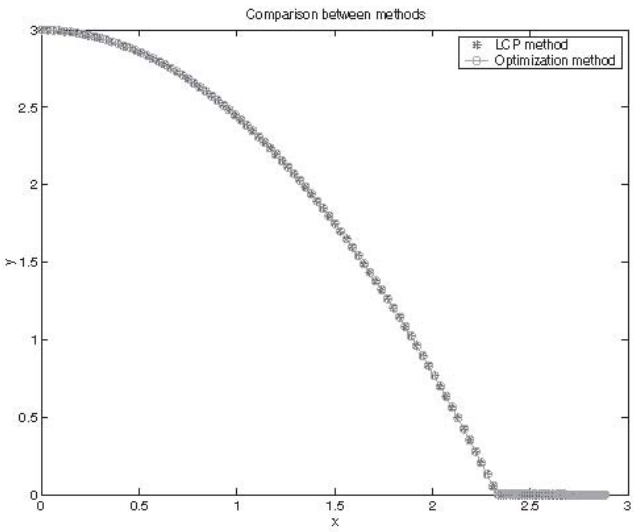
[see M.Anitescu, "Optimization Based Simulation of Nonsmooth Rigid Body Dynamics"]

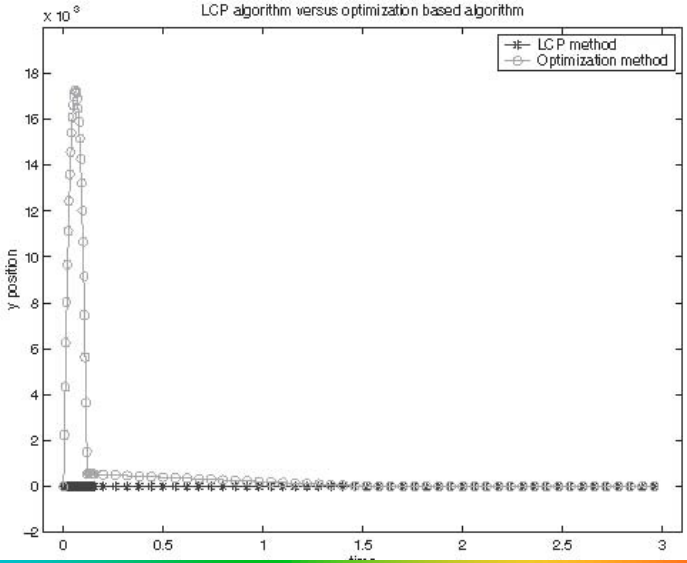
What is physical meaning of the relaxation?

- Origin



- Behavior





Cone complementarity

- Aiming at a more compact formulation:

$$
\begin{aligned}
\boldsymbol{b}_{\mathcal{A}} &= \left\{ \frac{1}{h}\Phi^{i_1}, 0, 0, \frac{1}{h}\Phi^{i_2}, 0, 0, \ldots, \frac{1}{h}\Phi^{i_{n_{\mathcal{A}}}}, 0, 0 \right\} \\
\boldsymbol{\gamma}_{\mathcal{A}} &= \left\{ \gamma_n^{i_1}, \gamma_u^{i_1}, \gamma_v^{i_1}, \gamma_n^{i_2}, \gamma_u^{i_2}, \gamma_v^{i_2}, \ldots, \gamma_n^{i_{n_{\mathcal{A}}}}, \gamma_u^{i_{n_{\mathcal{A}}}}, \gamma_v^{i_{n_{\mathcal{A}}}} \right\} \\
\boldsymbol{b}_{\mathcal{B}} &= \left\{ \frac{1}{h}\Psi^1 + \frac{\partial \Psi^1}{\partial t}, \frac{1}{h}\Psi^2 + \frac{\partial \Psi^2}{\partial t}, \ldots, \frac{1}{h}\Psi^{n_{\mathcal{B}}} + \frac{\partial \Psi^{n_{\mathcal{B}}}}{\partial t} \right\} \\
\boldsymbol{\gamma}_{\mathcal{B}} &= \left\{ \gamma_b^1, \gamma_b^2, \ldots, \gamma_b^{n_{\mathcal{B}}} \right\}
\end{aligned}
$$

$$D_{\mathcal{A}} = \left[D^{i_1} | D^{i_2} | \ldots | D^{i_{n_{\mathcal{A}}}} \right], \quad i \in \mathcal{A}(\boldsymbol{q}^l, \epsilon) \qquad D^i = \left[\boldsymbol{D}_n^i | \boldsymbol{D}_u^i | \boldsymbol{D}_v^i \right]$$

- $D_{\mathcal{B}} = \left[\nabla\Psi^{i_1} | \nabla\Psi^{i_2} | \ldots | \nabla\Psi^{i_{n_{\mathcal{B}}}} \right], \quad i \in \mathcal{G}_{\mathcal{B}}$

$$\boldsymbol{b}_{\mathcal{E}} \in \mathbb{R}^{n_{\mathcal{E}}} = \{ \boldsymbol{b}_{\mathcal{A}}, \boldsymbol{b}_{\mathcal{B}} \}$$

$$\boldsymbol{\gamma}_{\mathcal{E}} \in \mathbb{R}^{n_{\mathcal{E}}} = \{ \boldsymbol{\gamma}_{\mathcal{A}}, \boldsymbol{\gamma}_{\mathcal{B}} \}$$

$$D_{\mathcal{E}} = [D_{\mathcal{A}} | D_{\mathcal{B}}]$$

Cone complementarity

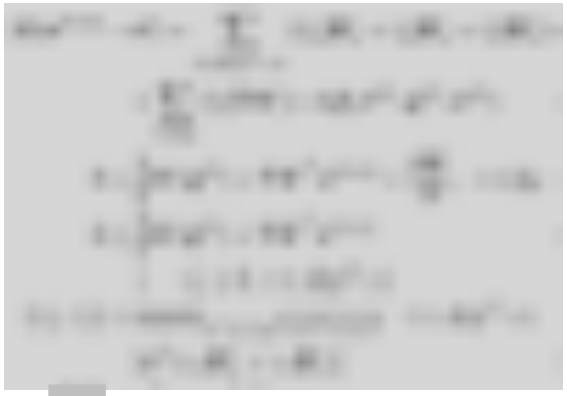
- Also define:

$$\tilde{\boldsymbol{k}}^{(l)} = M\boldsymbol{v}^{(l)} + h\boldsymbol{f}_t(t^{(l)}, \boldsymbol{q}^{(l)}, \boldsymbol{v}^{(l)})$$

$$N = D_{\mathcal{E}}^T M^{-1} D_{\mathcal{E}}$$

$$\boldsymbol{r} = D_{\mathcal{E}}^T M^{-1} \tilde{\boldsymbol{k}} + \boldsymbol{b}_{\mathcal{E}}$$

- Then:



becomes..

$$(N\boldsymbol{\gamma}_{\mathcal{E}} + \boldsymbol{r}) \in -\Upsilon^{\circ} \quad \perp \quad \boldsymbol{\gamma}_{\mathcal{E}} \in \Upsilon$$

This is a CCP,
CONE COMPLEMENTARITY PROBLEM

An iterative method- Step 3

- Convexification opens the path to high performance computing.
- How to efficiently solve the Cone Complementarity Problem for large-scale systems?

$$(N\boldsymbol{\gamma}_{\mathcal{E}} + \boldsymbol{r}) \in -\Upsilon^{\circ} \quad \perp \quad \boldsymbol{\gamma}_{\mathcal{E}} \in \Upsilon$$

- Our method: use a fixed-point iteration (Gauss-Seidel-Jacobi)

$$\boldsymbol{\gamma}^{r+1} = \lambda \Pi_{\Upsilon}\left(\boldsymbol{\gamma}^{r} - \omega B^{r}\left(N\boldsymbol{\gamma}^{r} + \boldsymbol{r} + K^{r}\left(\boldsymbol{\gamma}^{r+1} - \boldsymbol{\gamma}^{r}\right)\right)\right) + (1-\lambda)\,\boldsymbol{\gamma}^{r}$$

- with matrices:
- ..and a non-extensive separable projection operator onto feasible set

$$\Pi_{\Upsilon} : \mathbb{R}^{n_{\varepsilon}} \rightarrow \mathbb{R}^{n_{\varepsilon}}$$

$$B^{r} = \begin{bmatrix} \eta_1 I_{n_1} & 0 & \cdots & 0 \\ 0 & \eta_2 I_{n_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \eta_{n_k} I_{n_{n_k}} \end{bmatrix} \qquad K^{T} = \begin{bmatrix} 0 & K_{12} & K_{13} & \cdots & K_{1n_k} \\ 0 & 0 & K_{23} & \cdots & K_{2n_k} \\ 0 & 0 & 0 & \cdots & K_{3n_k} \\ \vdots & \vdots & \cdots & \ddots & \vdots \\ 0 & 0 & 0 & & 0 \end{bmatrix}$$

General: The iterative method

- ASSUMPTIONS

A1 The matrix N of the problem (CCP) is symmetric and positive semi-definite.

A2 There exists a positive number, $\alpha > 0$ such that, at any iteration r, $r = 0, 1, 2, \ldots$, we have that $B^r \succ \alpha I$

A3 There exists a positive number, $\beta > 0$ such that, at any iteration r, $r = 0, 1, 2, \ldots$, we have that $(x^{r+1} - x^r)^T \left((\lambda \omega B^r)^{-1} + K^r - \frac{N}{2} \right) (x^{r+1} - x^r) \geq \beta \left\| x^{r+1} - x^r \right\|^2$.

Always satisfied in multibody systems

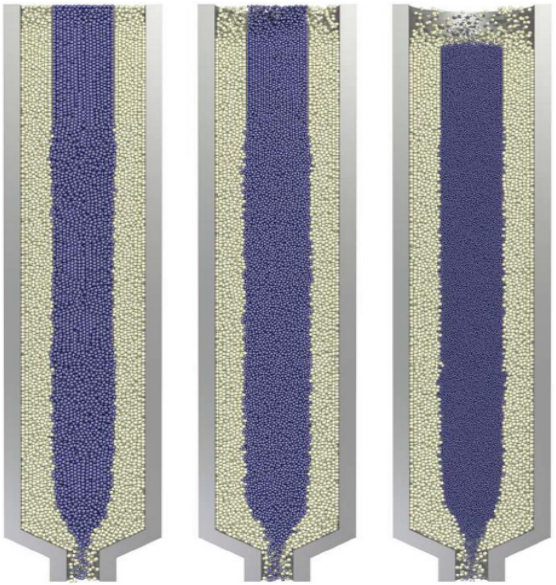
Essentially free choice, we use identity blocks

Use ω overrelaxation factor to adjust this

- Under the above assumptions, we can prove convergence.
- The method produces in absence of jamming a bounded sequence with an unique accumulation point
- Method is relatively easy to parallelize, even for GPU!

Challenge: simulating PBNR

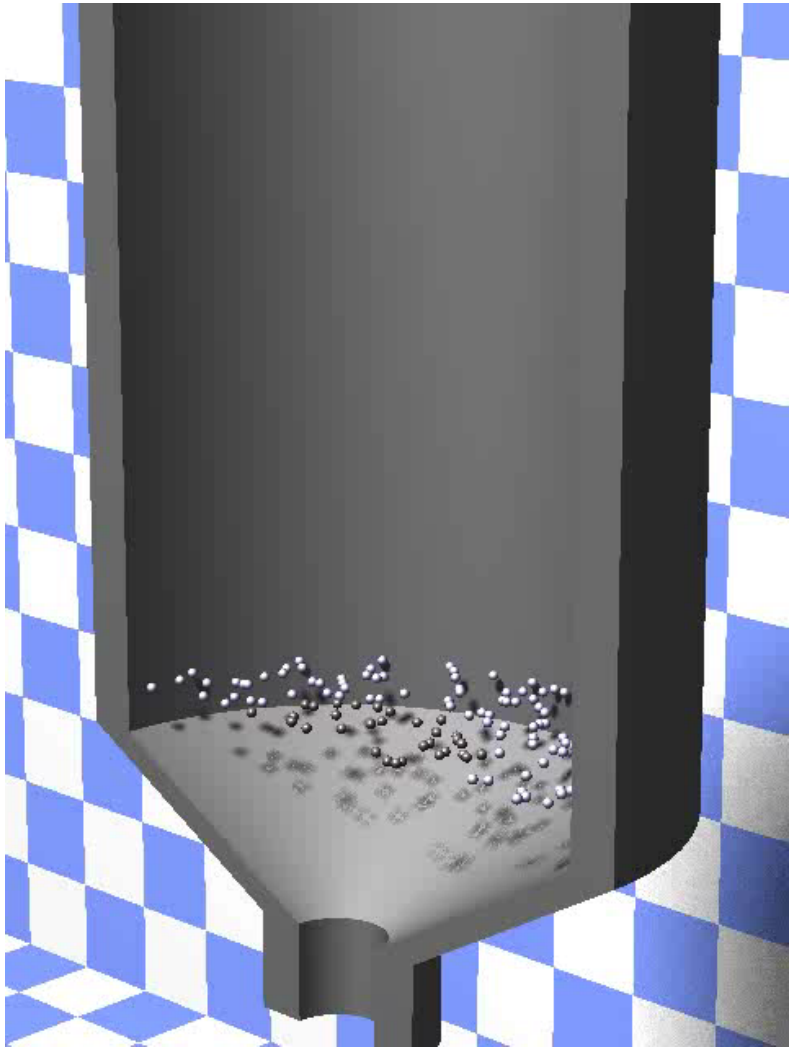
- Generation IV nuclear reactor with continuously moving fuel.
- Previous attempts: DEM methods on supercomputers at Sandia Labs regularization)
- 40 seconds of LAMMPS simulation for 440,000 pebbles needs 3 days on 64 processors dedicated cluster (Rycroft et al.)



model a frictionless wall, $\mu_w = 0.0$. For the current simulations we set $k_t = \frac{2}{7} k_n$ and choose $k_n = 2 \times 10^5$ gm/d. While this is significantly less than would be realistic for graphite pebbles, where we expect $k_n > 10^{10}$ gm/d, such a spring constant would be prohibitively computationally expensive, as the time step scales as $\delta t \propto k_n^{-1/2}$ for collisions to be modeled effectively. Previous simulations have shown that

Simulating the PBR nuclear reactor

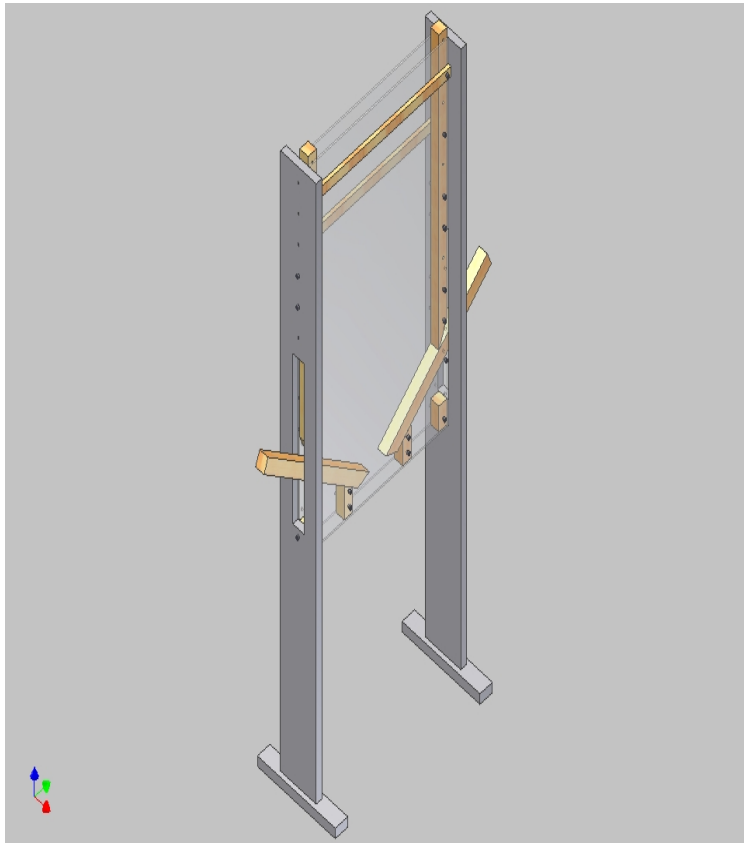
- 160’000 Uranium-Graphite spheres, 600’000 contacts on average
- One step: Two millions of primal variables, six millions of dual variables. 4000 0.01 ms steps.
- *1 day on a Windows station; shows linear performance*.
- We estimate (extrapolate) 3 CPU days, compare with 192 CPU days for smoothing (DEM-LAMMPS) under-resolved solution in 2006 !!!

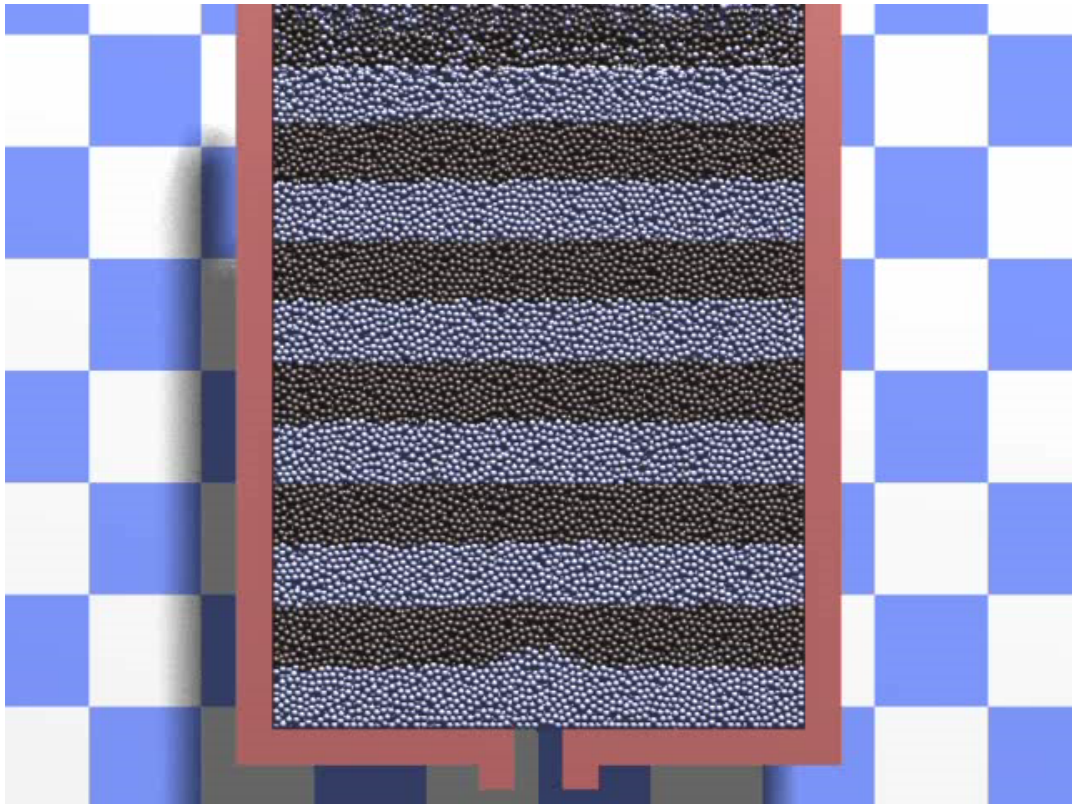


Validation

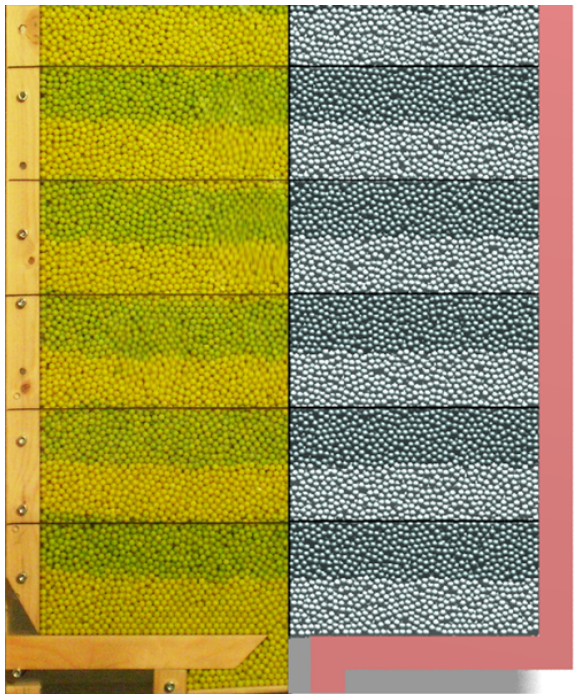
- In our experience, time-stepping would not have worked at this number of particles without the convex relaxation.
- Also, we are aware that our convergence result does not apply for nonzero restitution.
- Did we destroy the physics and the predictive power of the scheme?
- We believe not, at least in the dense granular flow case. Evidence, based on particle statistics:

DVI: Time-stepping validation : Hopper (Tasora & A 2009)

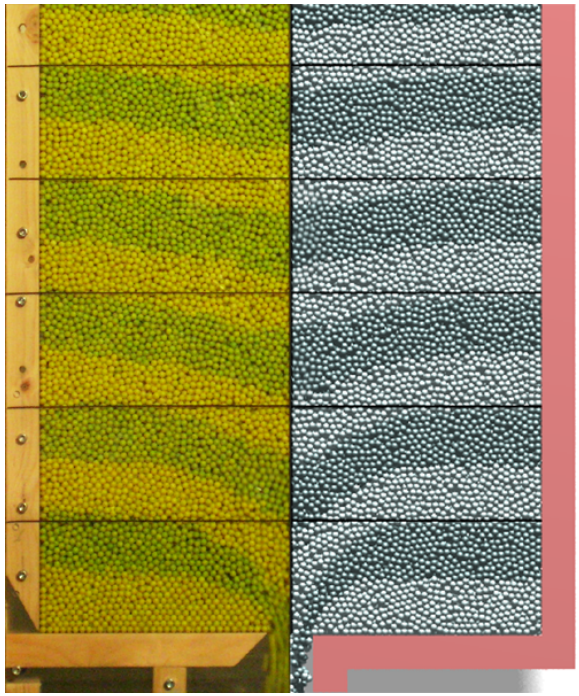




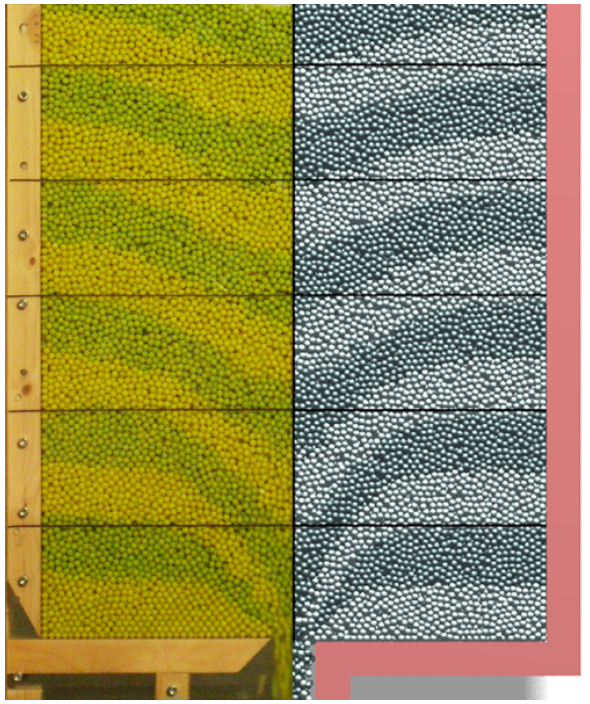
Hopper experiment and simulation: Images



t=0s

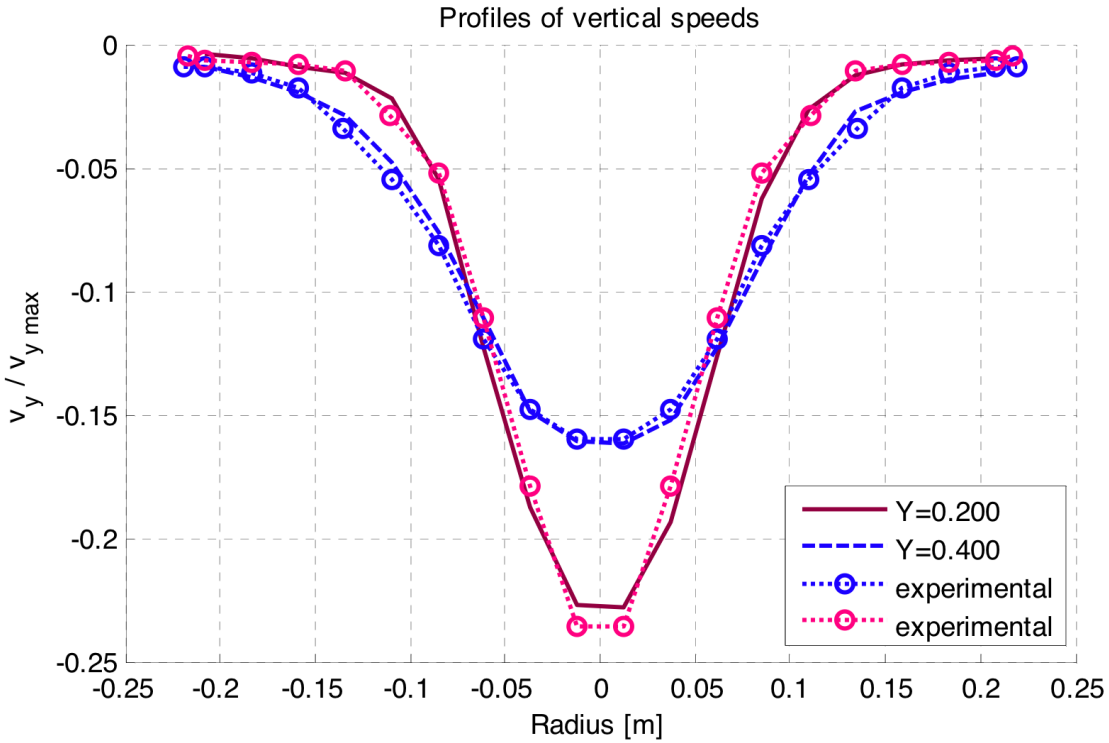


t=0.6s



t=1.2s

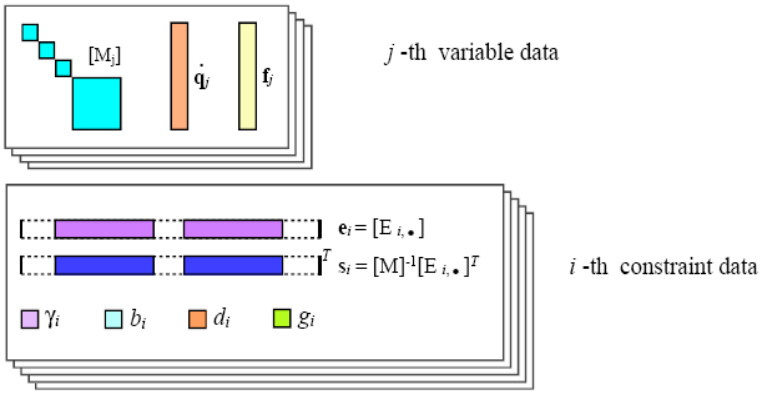
Hopper Results: Velocity (Tasora &A 2009)



- Note that there are sphere measurement errors of 2%, and particle-wall friction variations of 10% (reduced by climate control).
- So we declare validation a success for convex method.

The algorithm

- Development of an efficient algorithm for fixed point iteration:



- *avoid temporary data, exploit sparsity. Never compute explicitly the N matrix!*
- *implemented in incremental form. Compute only deltas of multipliers.*
- *O(n) space requirements and supports premature termination*

The algorithm is specialized, for minimum memory use!

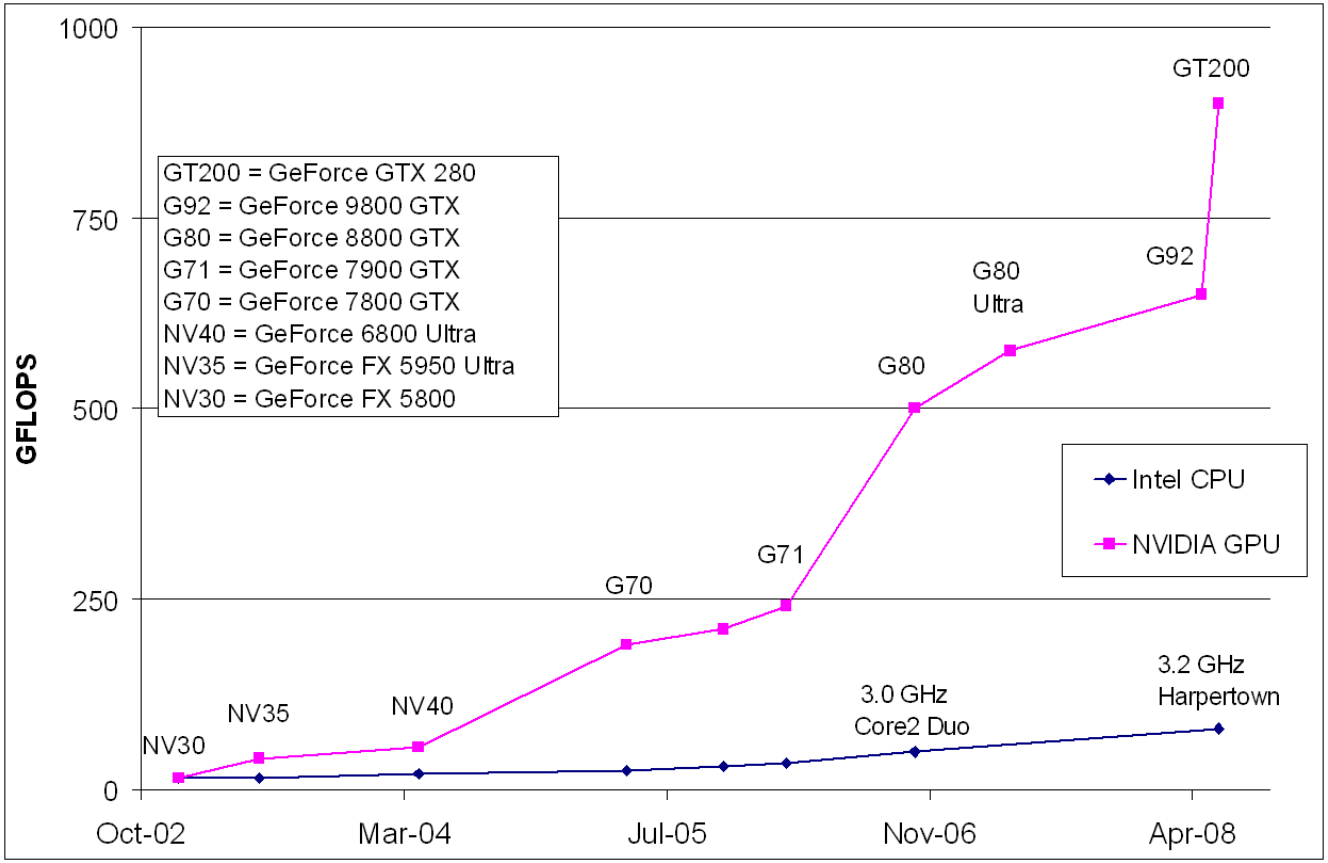
COM: Constraint <-> Body

(1) // Pre-compute some data for friction constraints
(2) for $i := 1$ to $n_{\mathcal{A}}$
(3) $\quad \boldsymbol{s}_a^i = M^{-1} D^i$
(4) $\quad g_a^i = D^{i,T} \boldsymbol{s}_a^i$
(5) $\quad \eta_a^i = \frac{3}{\mathrm{Trace}(g_a^i)}$
(6) // Pre-compute some data for bilateral constraints
(7) for $i := 1$ to $n_{\mathcal{B}}$
(8) $\quad s_b^i = M^{-1} \nabla \Psi^i$
(9) $\quad g_b^i = \nabla \Psi^{i,T} s_b^i$
(10) $\quad \eta_b^i = \frac{1}{g_b^i}$
(11)
(12) // Initialize impulses
(13) if warm start with initial guess $\boldsymbol{\gamma}_{\mathcal{E}}^*$
(14) $\quad \boldsymbol{\gamma}_{\mathcal{E}}^0 = \boldsymbol{\gamma}_{\mathcal{E}}^*$
(15) else
(16) $\quad \boldsymbol{\gamma}_{\mathcal{E}}^0 = 0$
(17)
(18) // Initialize speeds
(19) $\boldsymbol{v} = \sum_{i=1}^{n_{\mathcal{A}}} \boldsymbol{s}_a^i \boldsymbol{\gamma}_a^{i,0} + \sum_{i=1}^{n_{\mathcal{B}}} s_b^i \gamma_b^{i,0} + M^{-1} \tilde{\boldsymbol{k}}$

(21) // Main iteration loop
(22) for $r := 0$ to r_{max}
(23) $\quad$ // Loop on frictional constraints
(24) $\quad$ for $i := 1$ to $n_{\mathcal{A}}$
(25) $\quad\quad \boldsymbol{\delta}_a^{i,r} = \left(\boldsymbol{\gamma}_a^{i,r} - \omega \eta_a^i \left(D^{i,T} \boldsymbol{v}^r + \boldsymbol{b}_a^i\right)\right)$;
(26) $\quad\quad \boldsymbol{\gamma}_a^{i,r+1} = \lambda \Pi_{\Upsilon}\left(\boldsymbol{\delta}_a^{i,r}\right) + (1-\lambda)\boldsymbol{\gamma}_a^{i,r}$;
(27) $\quad\quad \Delta \boldsymbol{\gamma}_a^{i,r+1} = \boldsymbol{\gamma}_a^{i,r+1} - \boldsymbol{\gamma}_a^{i,r}$;
(28) $\quad\quad \boldsymbol{v} := \boldsymbol{v} + \boldsymbol{s}_a^{i\,T} \Delta \boldsymbol{\gamma}_a^{i,r+1}$.
(29) $\quad$ // Loop on bilateral constraints
(30) $\quad$ for $i := 1$ to $n_{\mathcal{B}}$
(31) $\quad\quad \delta_b^{i,r} = \left(\gamma_b^{i,r} - \omega \eta_b^i \left(\nabla \Psi^{i,T} \boldsymbol{v}^r + b_b^i\right)\right)$;
(32) $\quad\quad \gamma_b^{i,r+1} = \lambda \Pi_{\Upsilon}\left(\delta_b^{i,r}\right) + (1-\lambda)\gamma_b^{i,r}$;
(33) $\quad\quad \Delta \gamma_b^{i,r+1} = \gamma_b^{i,r+1} - \gamma_b^{i,r}$;
(34) $\quad\quad \boldsymbol{v} := \boldsymbol{v} + s_b^{i\,T} \Delta \gamma_b^{i,r+1}$.
(35)
(36) return $\boldsymbol{\gamma}_{\mathcal{E}}$, $\boldsymbol{v}$

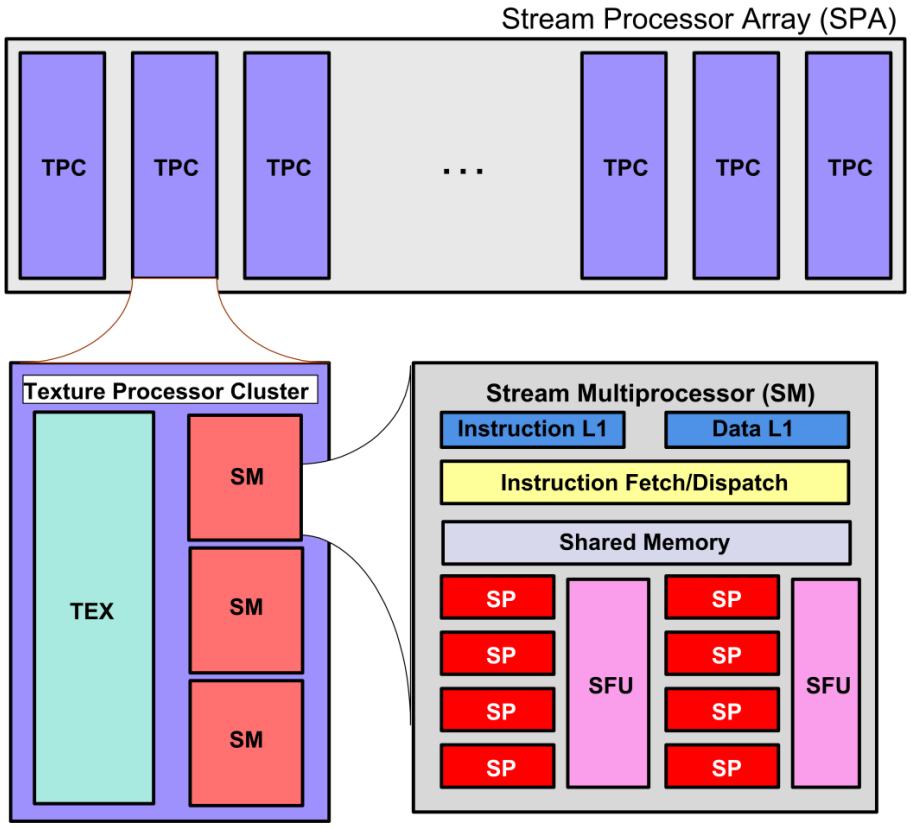
The rest “in place” per-body or per-constraint for Gauss-Jacobi

GPU : The attraction.



- Your PC graphic board is a supercomputer (0.32TF, GT8800).
- 5.7 TF: IBM BG/L $1,400K (2007) – NVIDIA Tesla $7K (2008)

NVIDIA TESLA C1060



- 30 Stream Multiprocessors.
- 240 Scalar Processors
- 4 GB device memory
- Memory Bandwidth: 102 GB/s
- Clock Rate: 1.3GHz
- Approx. $1,250

Parallel CCP on GPU: The 30,000 Feet Perspective

- Relies on a Gauss-Jacobi iteration: the first step.
- The GPU is viewed as a compute device that:
  - Is a co-processor to the CPU or host
  - Has its own DRAM (device memory)
  - Runs many threads in parallel (30K)
- Data-parallel portions (such as per-body "in-place") of an application are executed on the device as kernels which run in parallel on many threads
- Each simulation time step invokes multiple GPU calls
  - For each of these calls, parallelism can be on a
    - *"Per body" basis (work is done on different bodies in parallel)*
    - *"Per contact" basis (different contact events are processed in parallel)*

GPU: The CCP Pre-Processing

1. (*GPU, body-parallel*) **Force kernel**. For each body, compute applied external forces $\mathbf{f}(t^{(l)}, \mathbf{q}^{(l)}, \mathbf{v}^{(l)})$ (for example, gravitational and gyroscopic forces). Produce the force $\mathbf{F}_j$ and the torque $\mathbf{C}_j$ acting at CM of each body j.

2. (*GPU, contact-parallel*) **Contact preprocessing kernel**. For each contact i, given contact normal and position, compute in-place the matrices $\mathbf{D}^T_{i,v_A}$, $\mathbf{D}^T_{i,\omega_A}$ and $\mathbf{D}^T_{i,\omega_B}$, and the contact residual $\mathbf{b}_i = \{\frac{1}{h}\Phi_i(\mathbf{q}), 0, 0\}^T$.

3. (*GPU, body-parallel*) **Velocity Initialization kernel**. For each body j, initialize body velocity corrections: $\Delta\dot{\mathbf{r}}_j^{(l+1)} = h\ m_j^{-1}\mathbf{F}_j$ and $\Delta\omega_j^{(l+1)} = h\ \mathbf{J}_j^{-1}\mathbf{C}_j$.

GPU: The CCP Loop

4. (*GPU, contact-parallel*) **CCP iteration kernel**. For each contact i, do $\gamma_i^{r+1} = \lambda\ \Pi_{\Upsilon_i}\left(\gamma_i^r - \omega\eta_i\left(\mathbf{D}_i^T\mathbf{v}^r + \mathbf{b}_i\right)\right) + (1-\lambda)\gamma_i^r$. Store $\Delta\gamma_i^{r+1} = \gamma_i^{r+1} - \gamma_i^r$ in contact buffer. Compute updates to the velocities of the two connected bodies A and B (like $\Delta\dot{\mathbf{r}}_{A_i}^{(l+1)} = m_{A_i}^{-1}\mathbf{D}_{i,v_A}\Delta\gamma_i^{r+1}$, $\Delta\omega_{A_i}^{(l+1)} = \mathbf{J}_{A_i}^{-1}\mathbf{D}_{i,\omega_A}\Delta\gamma_i^{r+1}$), and store them in the reduction buffer.

5. (*GPU, reduction-slot-parallel*) **Run body-velocity reduction kernel**.

6. (*GPU, body-parallel*) **Body velocity updates kernel**. For each j body, add the cumulative velocity updates: $\dot{\mathbf{r}}_j^{(l+1)} = \dot{\mathbf{r}}_j^{(l)} + \Delta\dot{\mathbf{r}}_j^{(l+1)}$, and $\omega_j^{(l+1)} = \omega_j^{(l)} + \Delta\omega_j^{(l+1)}$.

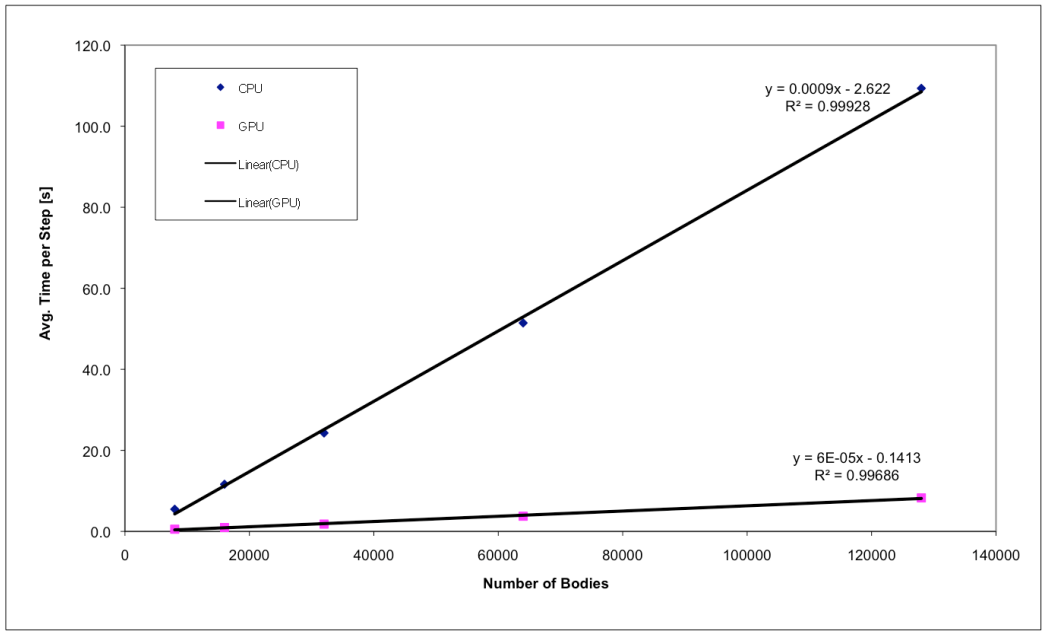
7. Repeat from step 4 until convergence or until number of CCP iterations reached $r > r_{max}$.

8. (*GPU, body-parallel*) **Time integration kernel**. For each j body, perform time integration as $\mathbf{q}_j^{(l+1)} = \mathbf{q}_j^{(l)} + h\mathbf{L}(\mathbf{q}_j^{(l)})\mathbf{v}_j^{(l+1)}$

9. (*CPU, serial*) If post processing required, fetch body data structures and contact multipliers from GPU memory to host memory.

GPU implementation: Reactor benchmark

- GPU implementation of the Anitescu-Tasora algorithm. (Tasora et al., (4)), NVIDIA with 16 stream processors.



	CPU			GPU					
Number of Bodies	CCP Time	CD Time	Step Time	CCP Time	CD Time	Step Time	CCP Speedup	CD Speedup	Step Speedup
128000	103.66658	3.80176	109.34287	6.97682	0.74488	8.27050	14.8587	5.1038	13.2208

Conclusions and future work.

- Granular dynamics is a topic of enormous practical importance.
- Time-stepping promises performance, stability and predictive power for granular dynamics.
- The convex relaxation works and is important for attaining many-million capability.
- The convex relaxation was VALIDATED

- Involving other physics such as fluid flow.
- Appropriate benchmarks for smoothing and time-stepping for large numbers of particles

Challenges and Open Questions.

- For large scale granular flow, can one solve the subproblem in O(N)?
- Can one define a successful multigrid approach for the subproblem?
- Given the conceptual connection between multigrid and homogenization, can one derive some form of continuum equations, at least for more regimes than known today?
- Multi-GPU dynamics algorithm implementation

What does convergence mean here ?

- Measure differential inclusion (Stewart 98)

$$M\frac{dv}{dt} - f_c(q,v) - k(t,q,v) \in FC(q).$$

Definition If ν is a measure and $K(\cdot)$ is a convex-set valued mapping, we say that v satisfies the differential inclusions

$$\frac{dv}{dt} \in K(t)$$

if, for all continuous $\phi \geq 0$ with compact support, not identically 0, we have that

$$\frac{\int \phi(t)\nu(dt)}{\int \phi(t)dt} \in \bigcup_{\tau:\phi(\tau)\neq 0} K(\tau).$$

Convergence result.

H1 The functions $n^{(j)}(q), t_1^{(j)}(q), t_2^{(j)}(q)$ are smooth and globally Lipschitz, and they are bounded in the 2-norm.

H2 The mass matrix M is positive definite.

H3 The external force increases at most linearly with the velocity and position.

H4 The uniform pointed friction cone assumption holds.

No Jamming !

Then there exists a subsequence $h_k \to 0$ where

- $q^{h_k}(\cdot) \to q(\cdot)$ uniformly.
- $v^{h_k}(\cdot) \to v(\cdot)$ pointwise a.e.
- $dv^{h_k}(\cdot) \to dv(\cdot)$ weak * as Borel measures. in [0,T], and every such subsequence converges to a solution $(q(\cdot), v(\cdot))$ of MDI.

Smoothing versus time-stepping

- Recall, DVI (for C=R+)

$$\dot{x} = f\left(t, x(t), u(t)\right);$$
$$u \geq 0 \perp F\left(t, x(t), u(t)\right) \geq 0$$

- Smoothing

$$\dot{x} = f\left(t, x(t), u(t)\right);$$
$$u_i \; F_i\left(t, x(t), u(t)\right) = \varepsilon, \quad i = 1, 2, \ldots n_u$$

- Followed by forward Euler. Easy to implement!!

$$u_i^n \; F_i\left(t^{n-1}, x^{n-1}, u^{n-1}\right) = \varepsilon, \quad i = 1, 2, \ldots n_u$$
$$x^{n+1} = x^n + hf\left(t^n, x^n, u^n\right);$$

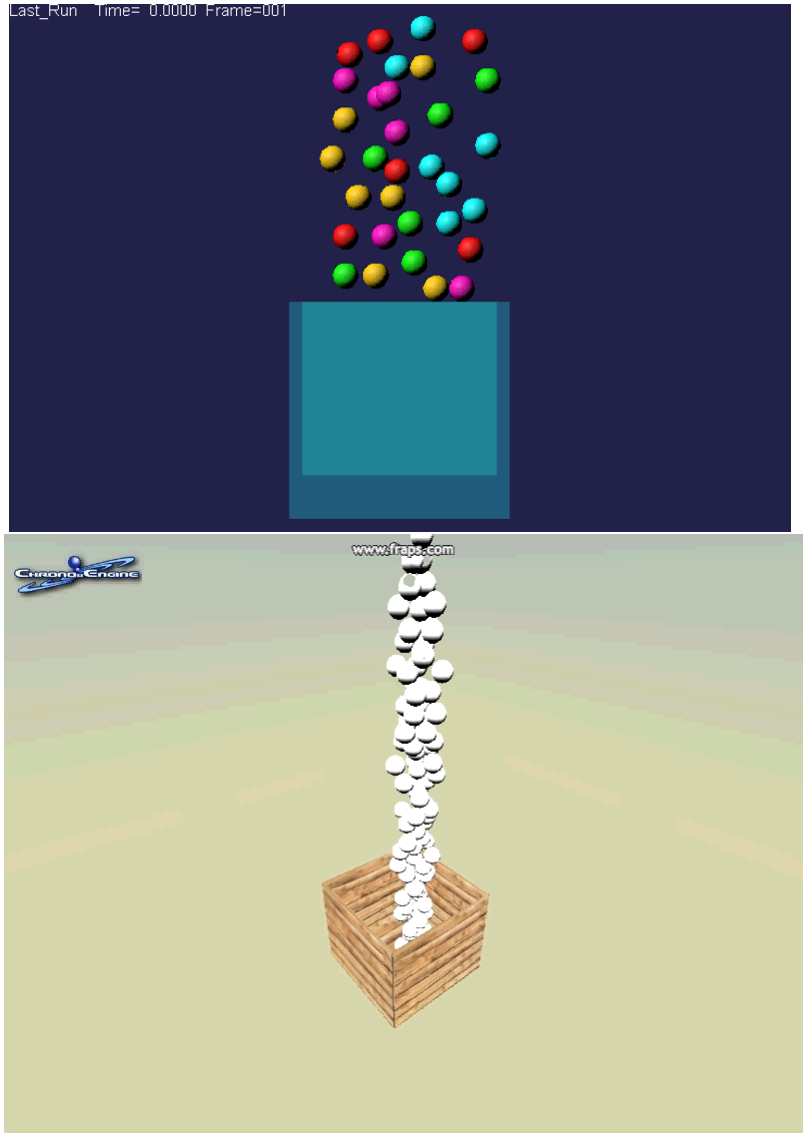
- Compare with the complexity of time-stepping

$$x^{n+1} = x^n + hf\left(t^{n+1}, x^{n+1}, u^{n+1}\right);$$
$$u^{n+1} \geq 0 \perp F\left(t^{n+1}, x^{n+1}, u^{n+1}\right) \geq 0$$

- But does it give good results?

Applying ADAMS to granular flow

- ADAMS is the workhorse of engineering dynamics.
- ADAMS/View Procedure for simulating.
- Spheres: diameter of 60 mm and a weight of 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



ADAMS versus ChronoEngine

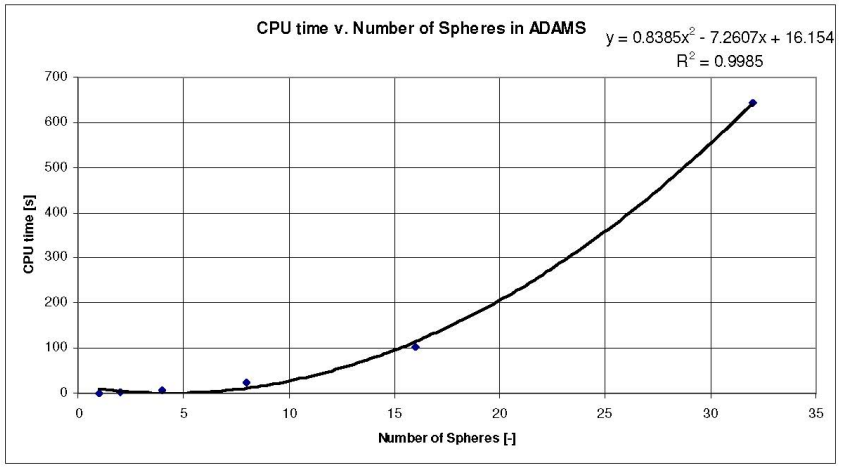
Table 1: Number of rigid bodies v. CPU time in ADAMS

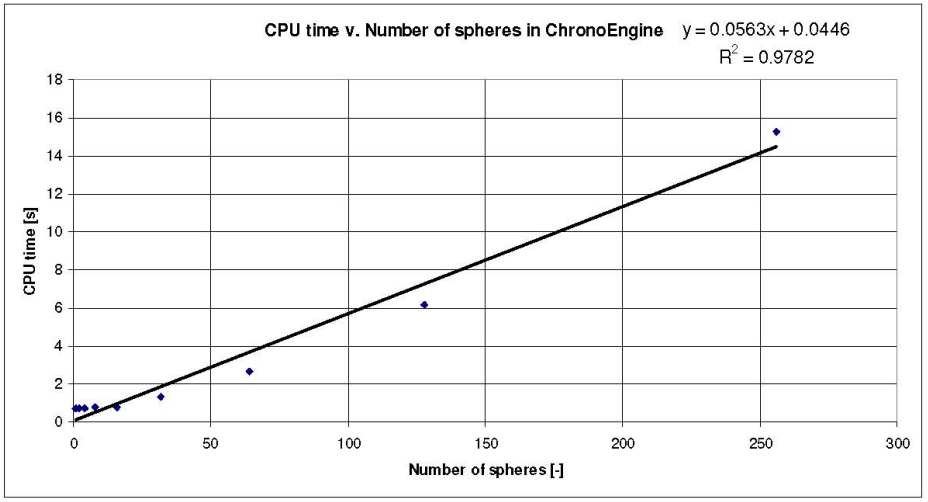
Number of Spheres	Max Number of Mutual Contacts [-]	CPU time (seconds)
1	1	0.41
2	3	3.3
4	14	7.75
8	44	25.36
16	152	102.78
32	560	644.4

The following graph shows the nonlinear increase in the CPU time as the number of colliding bodies increases.

Table 2: Number of rigid bodies v. CPU time in ChronoEngine

Number of Spheres	Max Number of Mutual Contacts [-]	CPU time (seconds)
1	1	0.70
2	3	0.73
4	14	0.73
8	44	0.76
16	152	0.82
32	560	1.32
64	2144	2.65
128	8384	6.17
256	33152	15.30



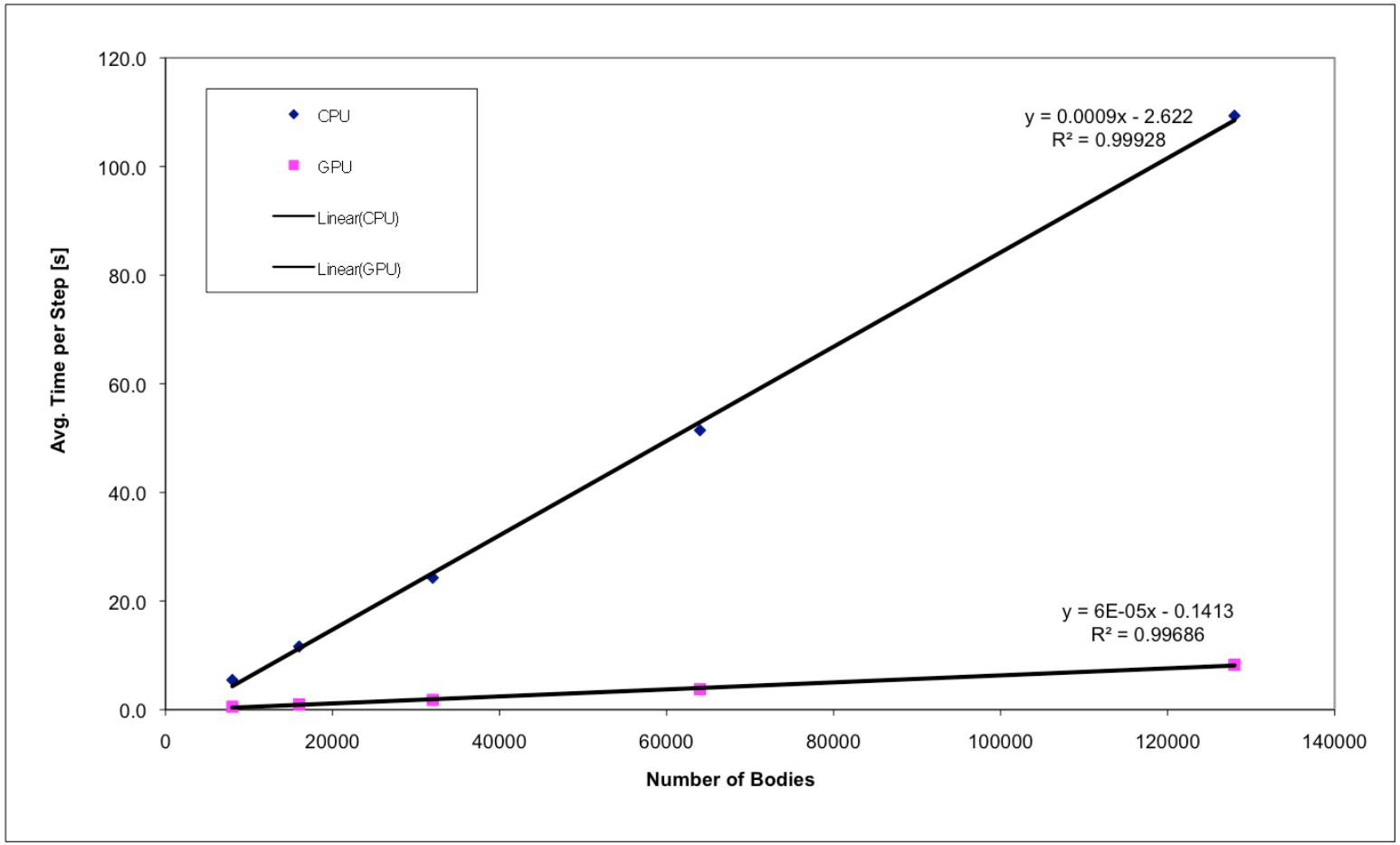


Often, time stepping is more promising. We follow this direction.

Performance

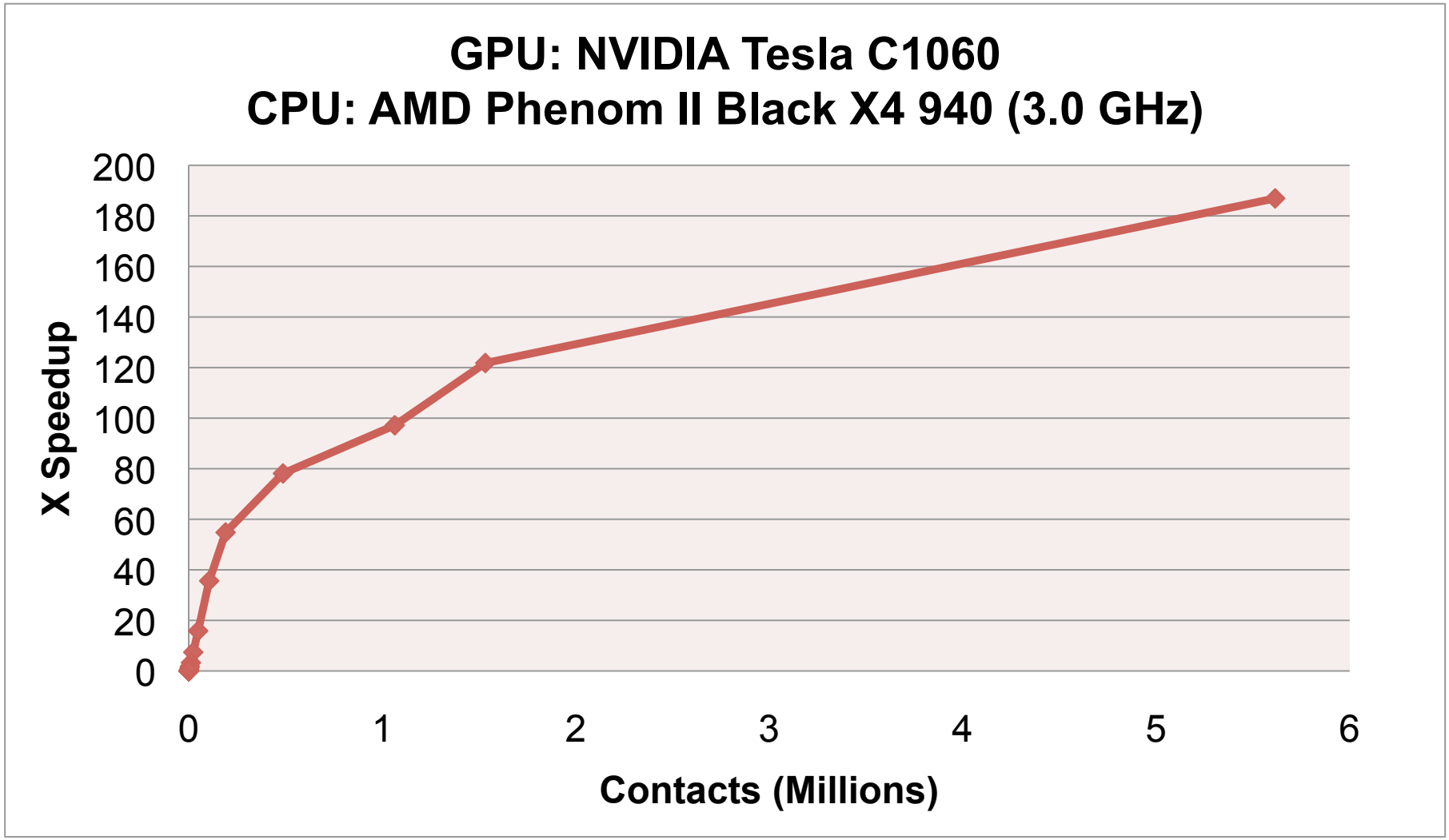
- Things are very much in flux.
- Subject (granular dynamics) is old, but there are no clear large scale computational benchmarks, since the concern was “Can one do it at all”.
- Our focus has been porting to GPU and various applications, and parameter choices are far from stable.
- This makes much harder toe-to-toe comparison with smoothing methods, though for small configurations, time-stepping advantage is clear (see ADAMS).
- Our experience (such as the PBR) suggest that we get, at least for GS, a factor of 50 reduction in effort due to the method for up to 1 mill particles, but we must test it for more and larger configurations.

PBR: GPU performance (Negrut et al, 2009)



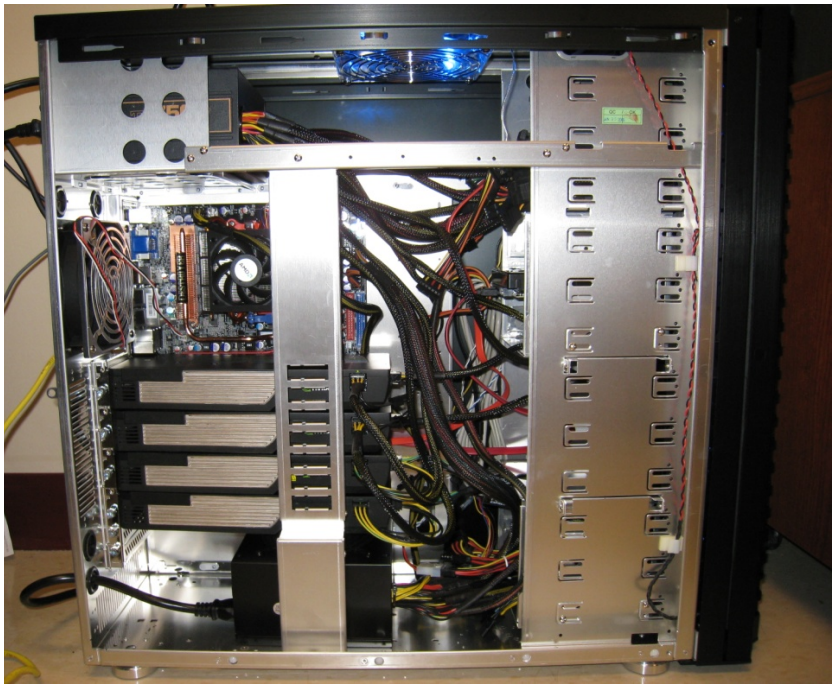
- It scales, but we still need “time-to-solution” comparisons between the various methods.

Speedup - GPU vs. CPU (Bullet library)



Multi-GPU Collision Detection

Assembled Quad GPU Machine



Processor: AMD Phenom II X4 940 Black

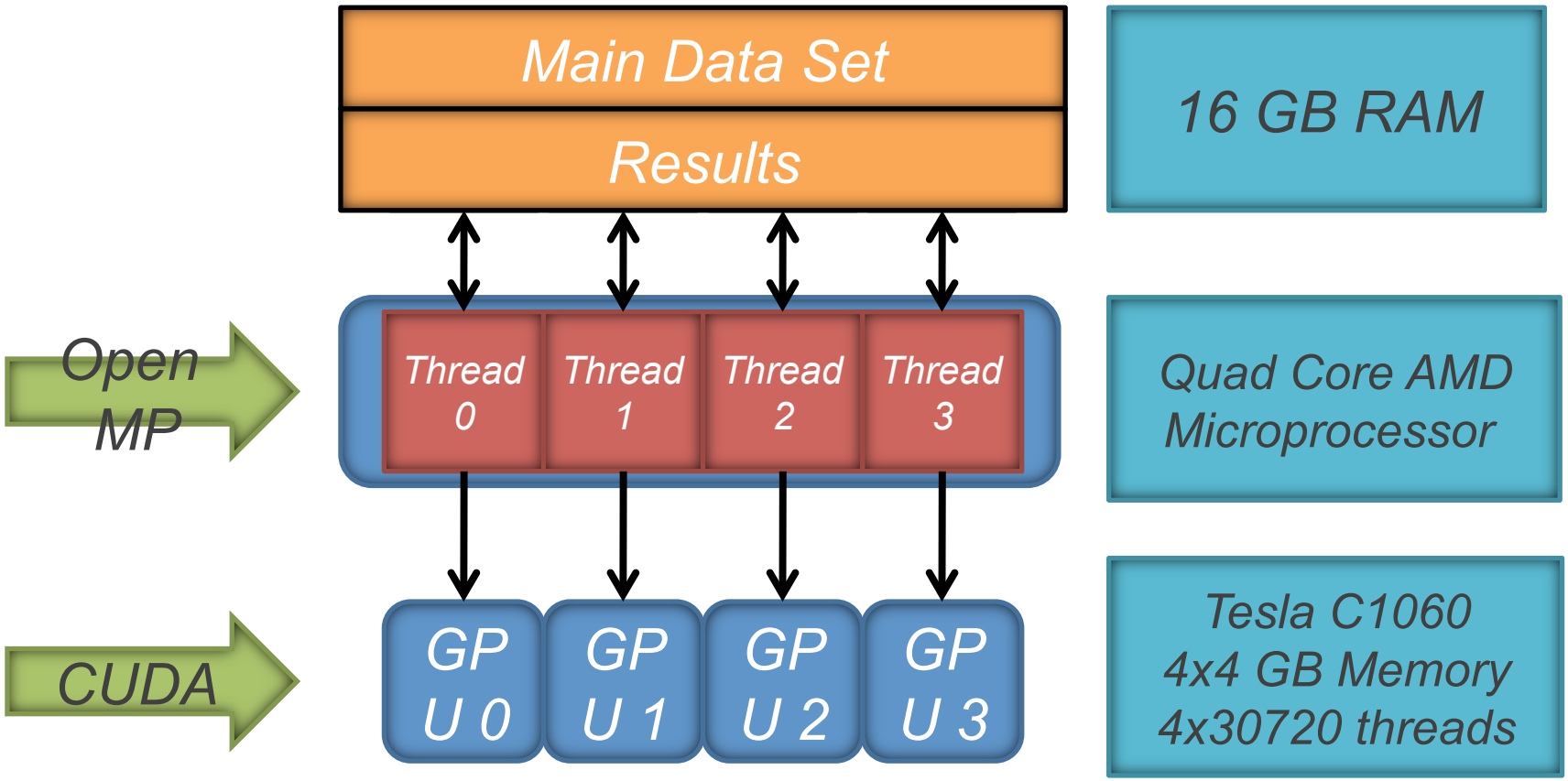
Memory: 16GB DDR2

Graphics: 4x NVIDIA Tesla C1060

Power supply 1: 1000W

Power supply 2: 750W

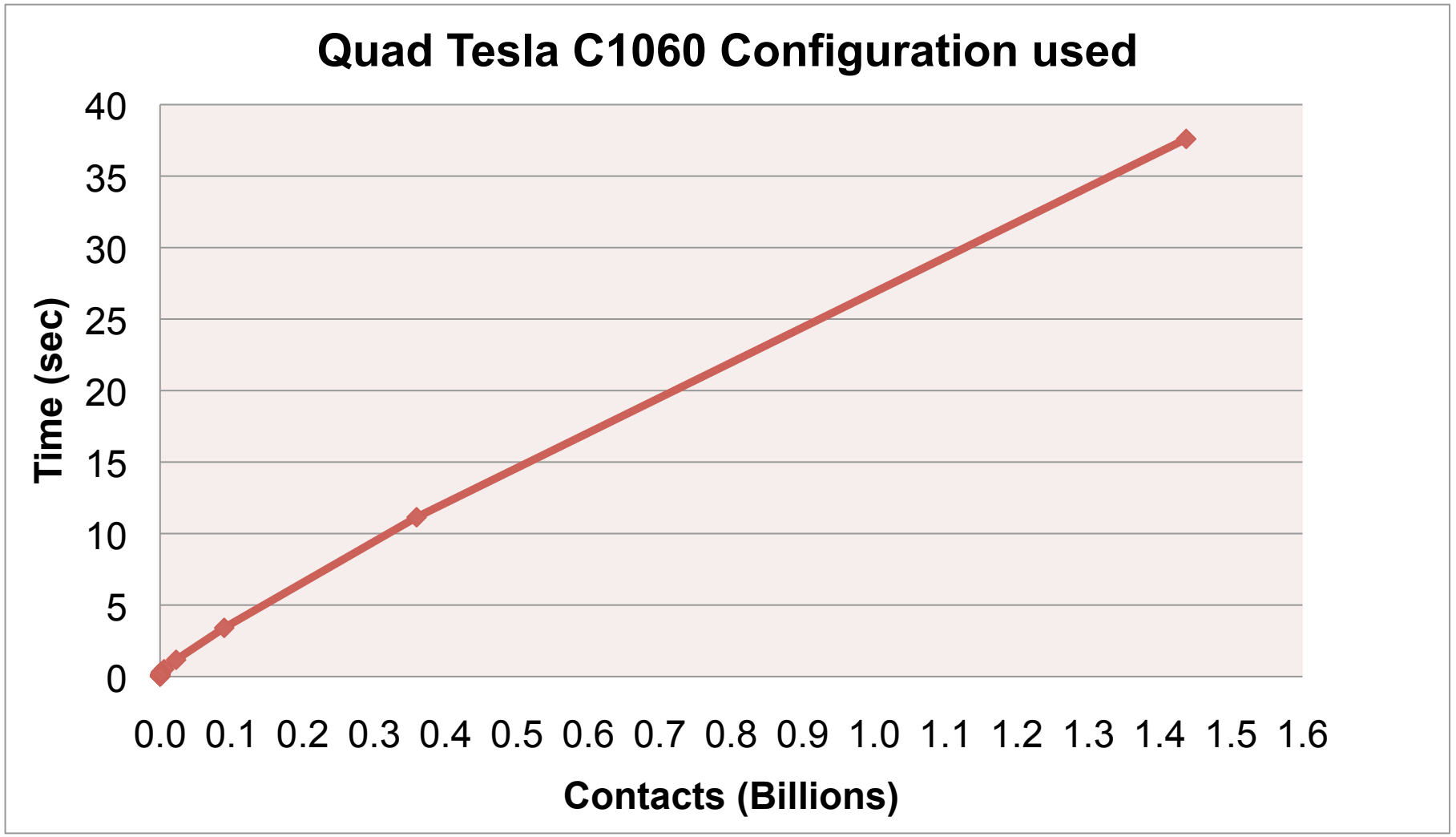
Processing Overview



Multi-GPU Collision Detection

- Split and organize data into Chunks
- Relying on OpenMP threads, one for each GPU
- Divide chunks into groups, GPUs work on chunk after chunk
- Combine collision data per group
- Combine collision data for all groups

Results – Contacts vs. Time (0.5 billion bodies)

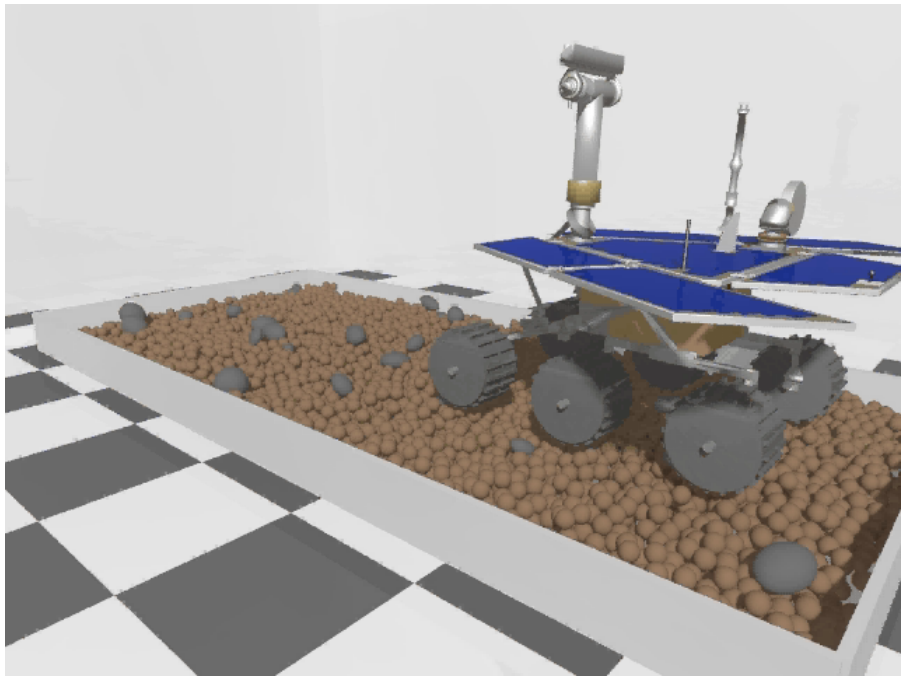


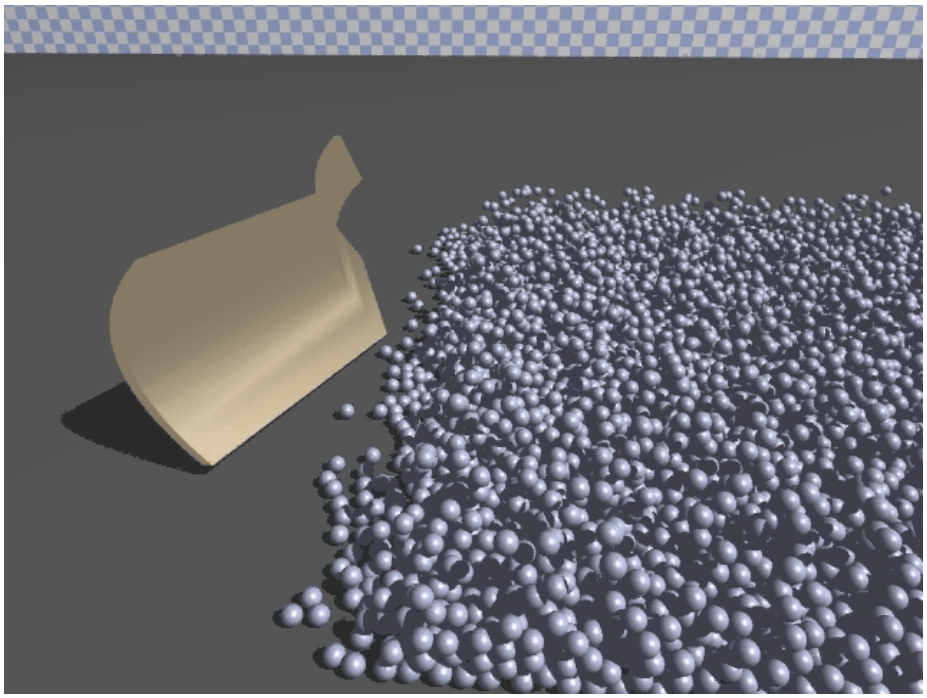
Performance: Conclusions

- It works: We currently run 1 million particles for 40 seconds in a few hours on a 2.66Ghs Intel GPU with a C1060 NVIDIA Tesla GPU (about 30K/1000K threads). Collision detection ~ 0.5 bill bodies on hybrid - 4GPU.
- In 06, LAMMPS-DEM was doing 400K particles under-resolved in 3 days on a 64 node cluster.
- In terms of capability per equipment dollars, progress is obvious – perhaps best HPC angle on GPU.
- Algorithmically, we need to establish relevant benchmarks to substantiate our “about 50 times improvement is due to the algorithm”.
  - With what smoothing settings do we run LAMMPS-DEM?
  - Which statistics do we consider? Etc.

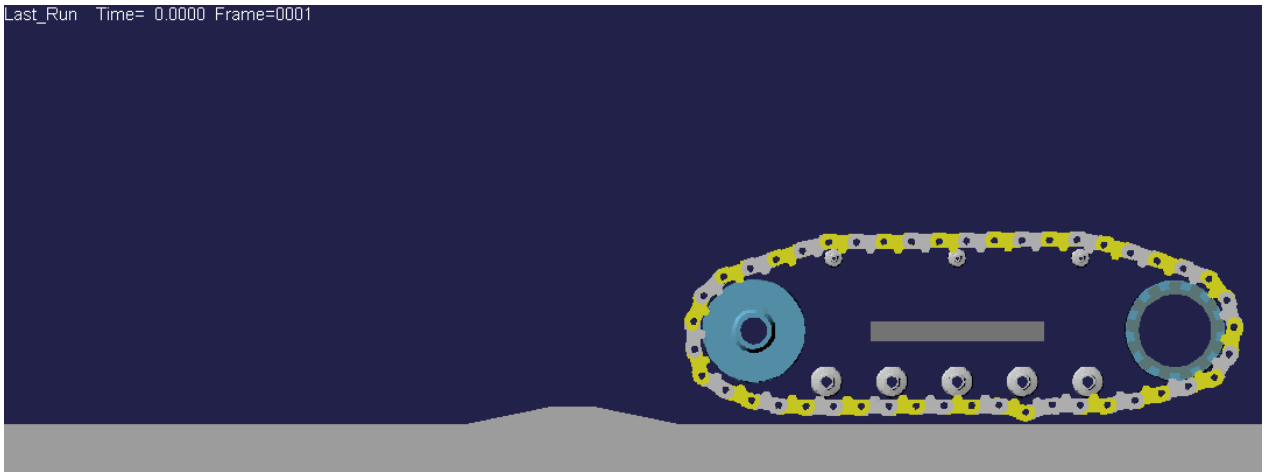
Some capabilities now available in our/your office: granular flow.

Some capabilities now available in your office: Vehicle Design





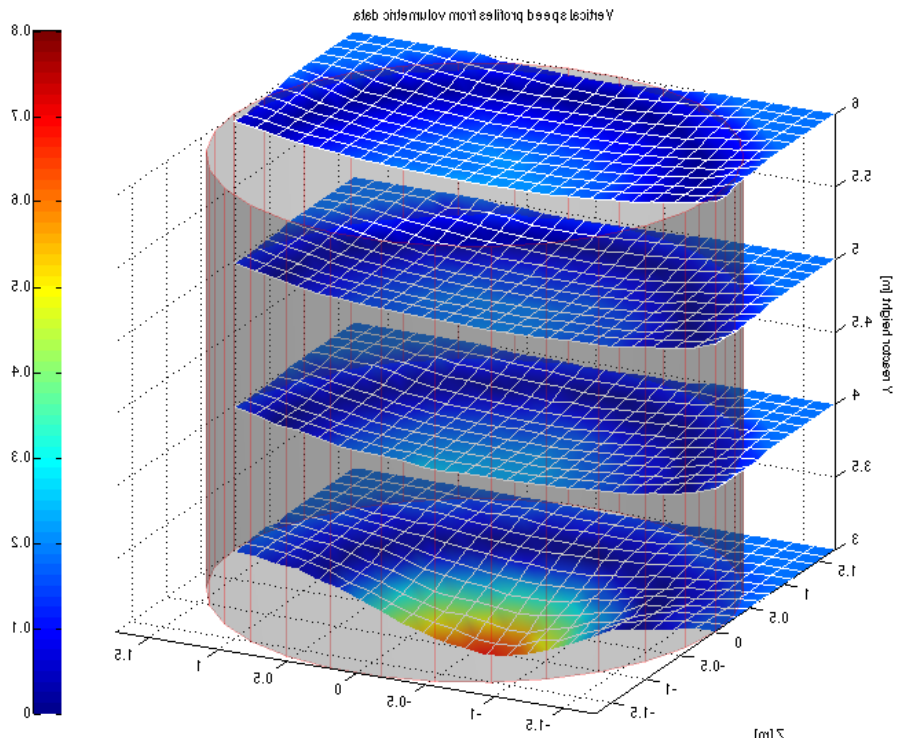
Some capabilities available in your office: tracked vehicle simulation

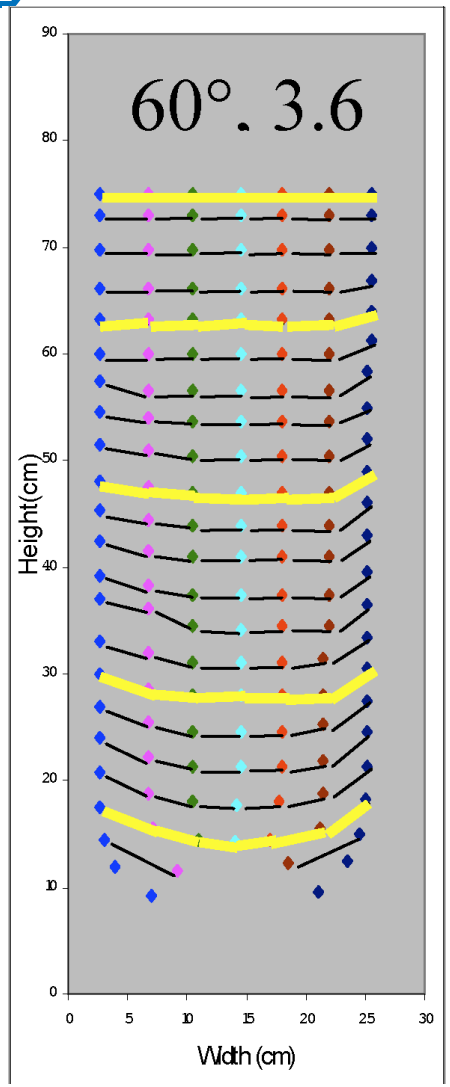


Thanks

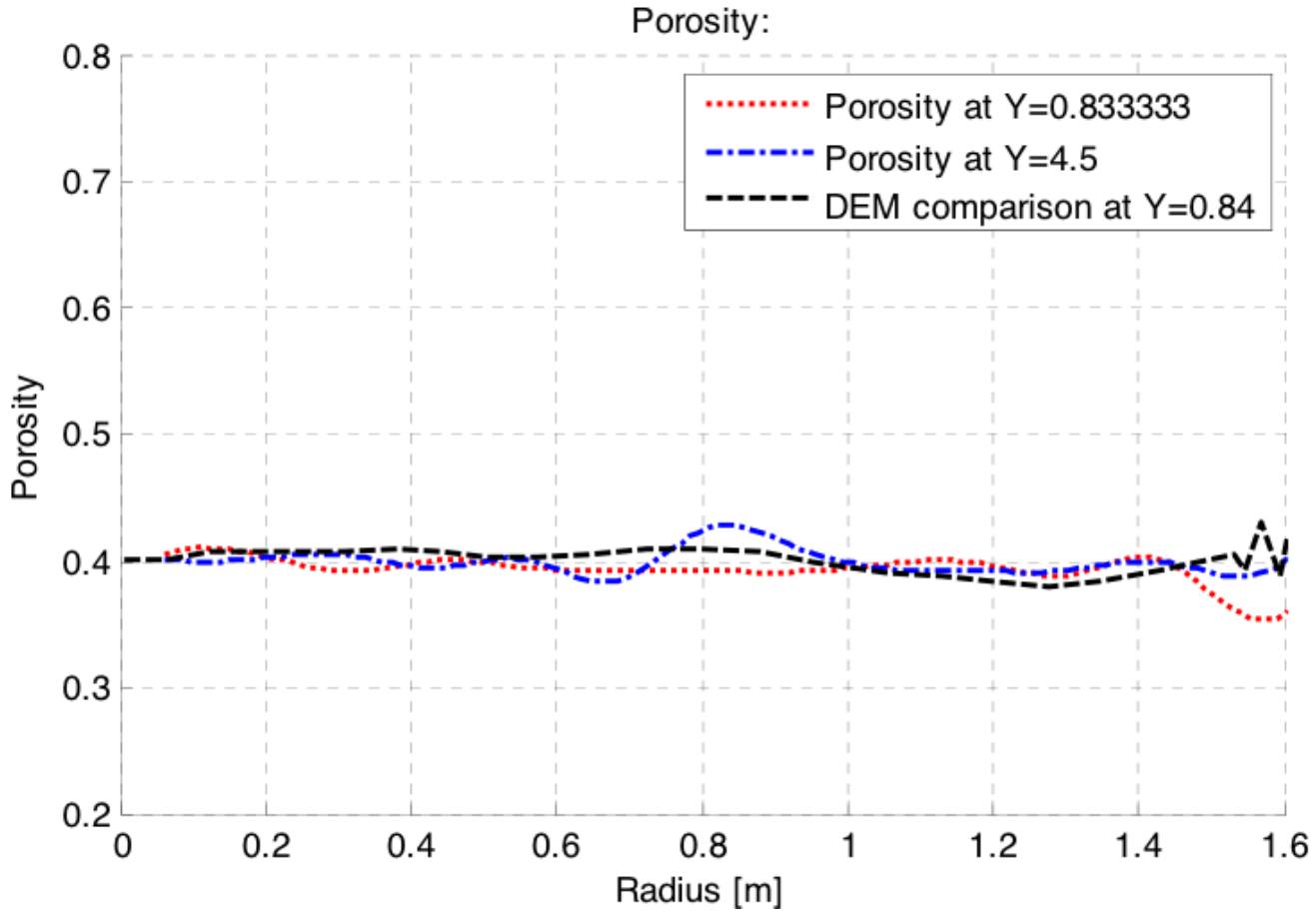
- Alessandro Tasora
  - University of Parma,
- Dan Negrut,
  - University of Wisconsin
- Department of Energy, Office of science Applied Math Program.
- NSF

Validation of convex relaxation time-stepping: PBR (Tasora &A 09_)





Validation: PBR: Packing statistics (Tasora & A, 09)



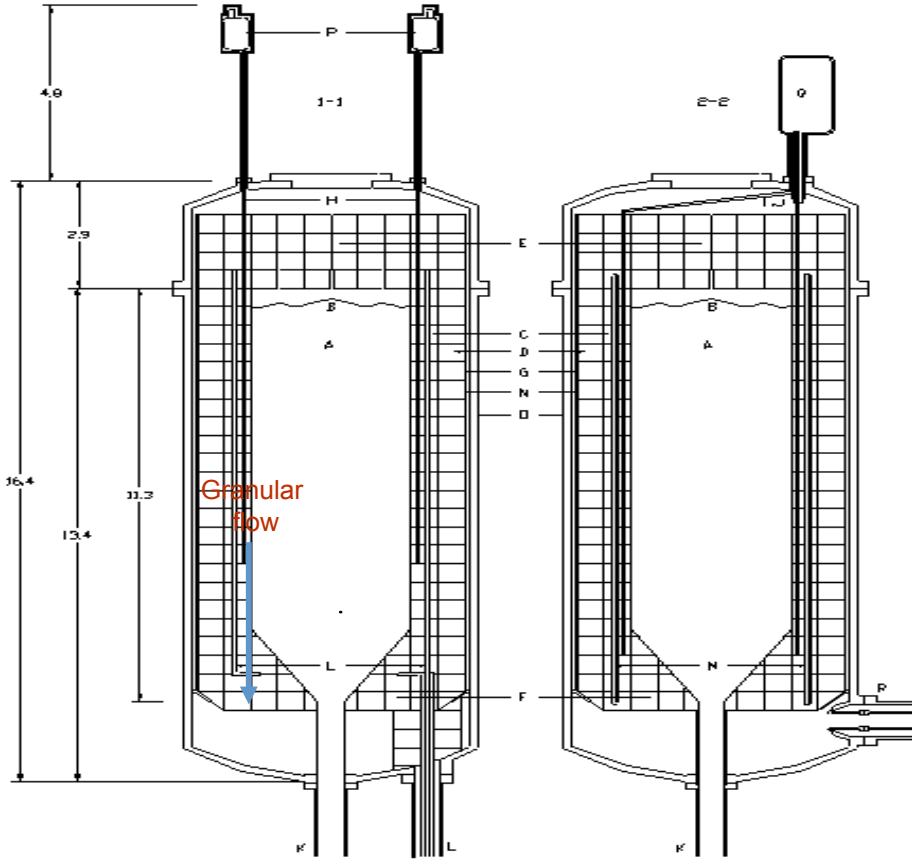
- The reactor is not far from “random” jamming. (VF 0.6)

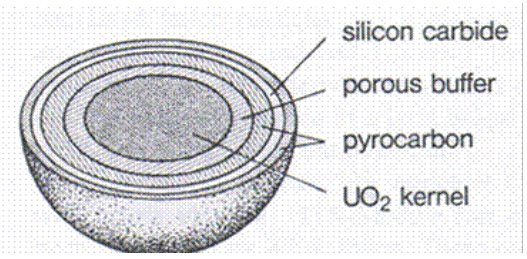
Short history and taxonomy of no-smoothing-methods in granular dynamics.

- Piecewise DAE (Haug, 86)
  - Plus : Uses well understood DAE technology
  - Minus: The density of switches, switching consistency, and Painleve (discontinuous velocity) are problems.
- Acceleration-force time-stepping (Glocker & Pfeiffer, 1992, Pang & Trinkle, 1995). Piecewise DAE + complementarity for switching.
  - Plus: No consistency problem.
  - Minus: Density of switches and Painleve.
- Velocity-impulse time-stepping. (Moreau, 196*, Stewart and Trinkle, 1996, A & Potra, 1997). Weak convergence: MDI
  - Plus: No consistency, or Painleve. Some have fixed time stepping (Moreau, 198*, Anitescu & Hart 04, Anitescu, 06).
  - Minus: Nonzero restitution coefficient is tough—but its value is disputable in any case

PBNR: The pebble bed nuclear reactor

- The PBNR nuclear reactor:
  - -Fourth generation design
  - -Inherently safe, by Doppler broadening of fission cross section
  - -Helium cooled > 1000 °C
  - -Can crack water (mass production of hydrogen)
  - -Continuous cycling of 400,000+ graphite spheres in a pebble bed.
  - -Question. Does it work *OK*?

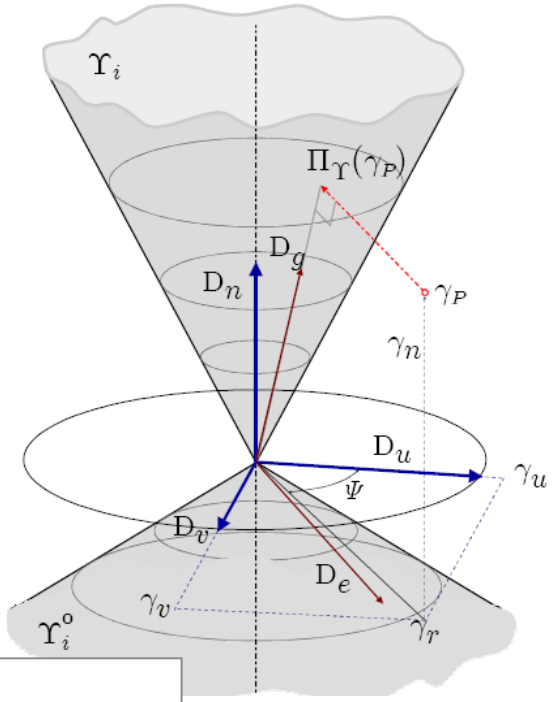




The projection operator is easy to compute and separable

- *For each frictional contact constraint:*

$$\Pi_\Upsilon = \left\{ \Pi_{\Upsilon_1}(\boldsymbol{\gamma}_1)^T, \ldots \Pi_{\Upsilon_{n_\mathcal{A}}}(\boldsymbol{\gamma}^{n_\mathcal{A}})^T, \Pi_b^1(\gamma_b^1), \ldots, \Pi_b^{n_\mathcal{B}}(\gamma_b^{n_\mathcal{B}}) \right\}^T$$



- *For each bilateral constraint, simply do nothing.*
- The complete operator:

$$\forall i \in \mathcal{A}(\boldsymbol{q}^{(l)}, \epsilon)$$

$$\begin{aligned}
\gamma_r < \mu_i \gamma_n \quad & \Pi_i = \boldsymbol{\gamma}_i \\
\gamma_r < -\frac{1}{\mu_i}\gamma_n \quad & \Pi_i = \{0,0,0\} \\
\gamma_r > \mu_i \gamma_n \wedge \gamma_r > -\frac{1}{\mu_i}\gamma_n \quad & \Pi_{i,n} = \frac{\gamma_r \mu_i + \gamma_n}{\mu_i^2 + 1} \\
& \Pi_{i,u} = \gamma_u \frac{\mu_i \Pi_{i,n}}{\gamma_r} \\
& \Pi_{i,v} = \gamma_v \frac{\mu_i \Pi_{i,n}}{\gamma_r}
\end{aligned}$$

Cone complementarity—Decomposable cones.

- Here we introduced the convex cone

$$\Upsilon = \left(\bigoplus_{i \in \mathcal{A}(\boldsymbol{q}^l, \epsilon)} \mathcal{FC}^i \right) \bigoplus \left(\bigoplus_{i \in \mathcal{G}_{\mathcal{B}}} \mathcal{BC}^i \right)$$

$\mathcal{BC}^i$ is R

In R^3 is i-th friction cone

- ..and its polar cone:

$$\Upsilon^\circ = \left(\bigoplus_{i \in \mathcal{A}(\boldsymbol{q}^l, \epsilon)} \mathcal{FC}^{i\circ} \right) \bigoplus \left(\bigoplus_{i \in \mathcal{G}_{\mathcal{B}}} \mathcal{BC}^{i\circ} \right)$$

CCP:

$$(N\boldsymbol{\gamma}_{\mathcal{E}} + \boldsymbol{r}) \in -\Upsilon^\circ \quad \perp \quad \boldsymbol{\gamma}_{\mathcal{E}} \in \Upsilon$$

Extensions: Nonlinear Model Predictive Control NLMPC (Zavala and A, 09)

- Parametric Optimization, such as Nonlinear Model Predictive Control, is One Particular Case of DVI.
- NLMPC is used in energy, petrochemical, chemical …
- Here, the VI are the optimality conditions of the NLMPC parametric optimization problem. $\min_{z \in C} f(z,s),\ \text{s.t.}\ g(z,s) = 0$

$$
\begin{array}{lcll}
y' & = & f(t, y(t), x(t)) & s' = 1 \\
x(t) & \in & SOL(K; F(t, y(t), \cdot)) & x(t) \in SOL(K; F(t, s, \cdot)) \\
y(0) & = & y_0 & s(0) = 0
\end{array}
$$

- Extension from DVI: Time-stepping, which solves one linear VI per step inexactly, converges to the NLMPC solution. Important for real-time implementation.
- Also solved by projected Gauss-Seidel (and Aug Lag).

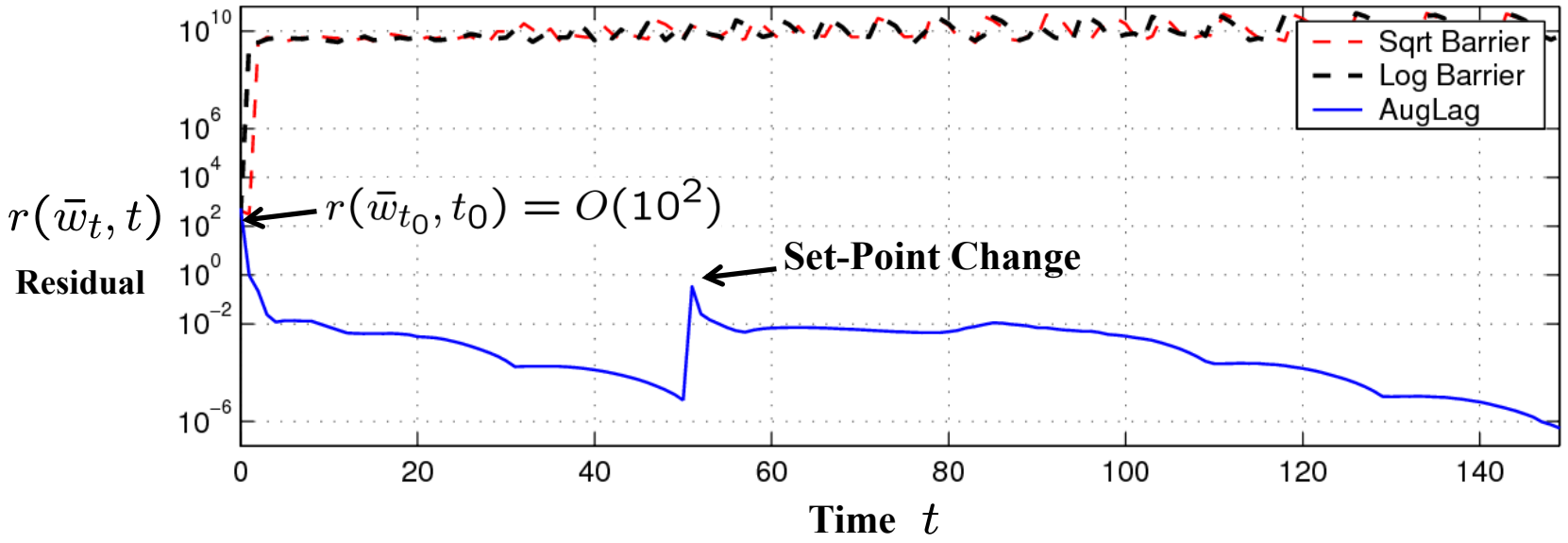
Numerical Study: Polymerization Reactor NLMPC

Numerical Tests

- Compare Against Smoothing *Heath, 2004, Ohtsuka, 2004*

1) $\mu \cdot \log(x - x^{min}) + \mu \cdot \log(x^{max} - x)$ **2)** $\mu \cdot \mathrm{sqrt}(x - x^{min}) + \mu \cdot \mathrm{sqrt}(x^{max} - x)$

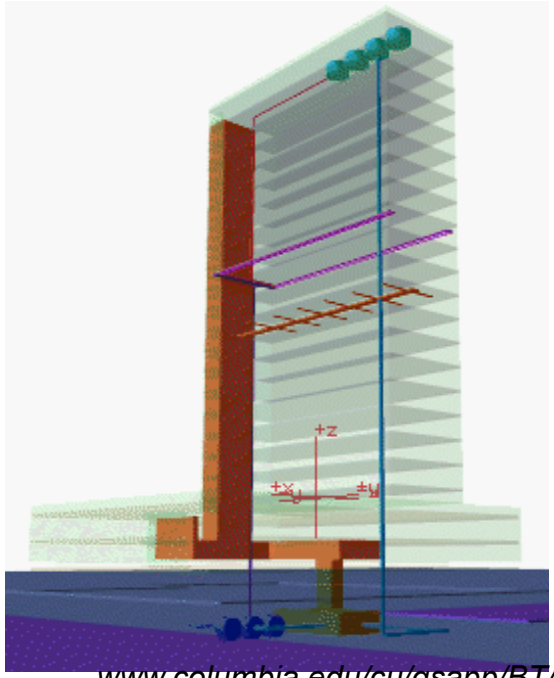
- $n_{PGS} = 25,\ \Delta t = 0.025,\ \rho = 100$



Smoothing is Numerically Unstable

AL Time-Stepping Stands Relatively Large Initial Errors

Thermal management of buildings (Zavala et al., 09)



www.columbia.edu/cu/gsapp/BT/LEVER/

Minimize Annual Heating and Cooling Costs

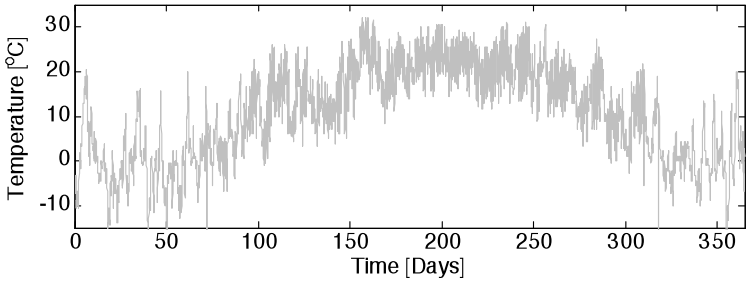
$$\min_{u(t)} \int_{t_\ell}^{t_\ell+N} \left[C_c(t)\varphi_c(t) + C_h(t)\varphi_h(t)\right] dt$$

$$\begin{aligned}
C_I \cdot \frac{\partial T_I}{\partial \tau} &= \varphi_h(\tau) - \varphi_c(\tau) - S \cdot \alpha' \cdot (T_I(\tau) - T_W(\tau, 0)) \\
\frac{\partial T_W}{\partial \tau} &= \beta \cdot \frac{\partial^2 T_W}{\partial x^2} \\
\alpha' \left(T_I(\tau) - T_W(\tau, 0)\right) &= -\mathbf{k} \cdot \left.\frac{\partial T_W}{\partial x}\right|_{(\tau,0)} \\
\alpha'' \left(T_W(\tau, L) - T_A(\tau)\right) &= -\mathbf{k} \cdot \left.\frac{\partial T_W}{\partial x}\right|_{(\tau,L)} \\
T_I(0) &= T_I^\ell \\
T_W(0, x) &= T_W^\ell(x)
\end{aligned}$$

Energy Balances

NLP with 100,000 Constraints & 20,000 Degrees of Freedom

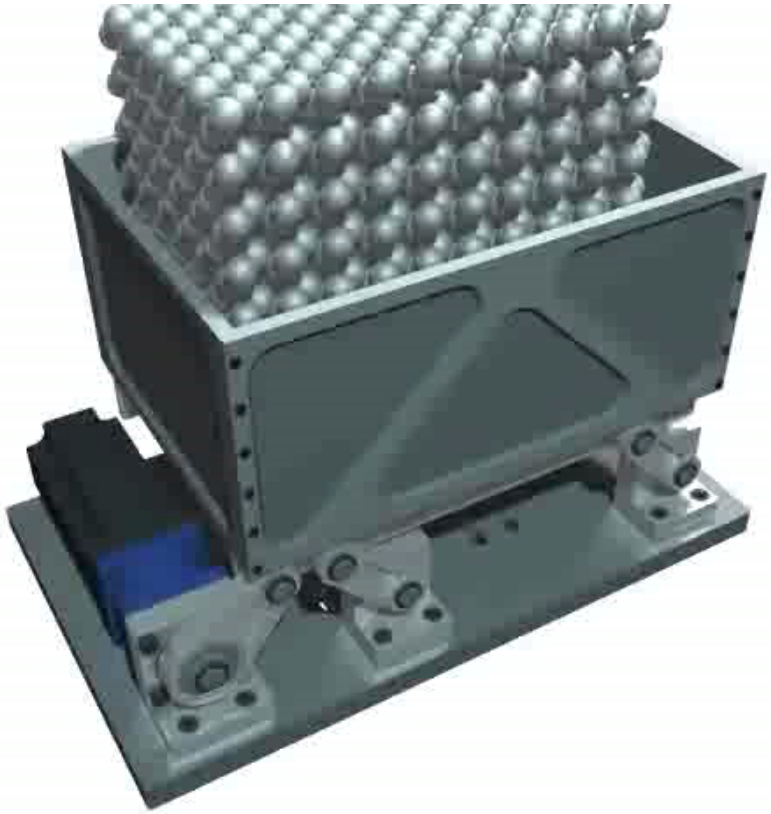
Time-Varying Electricity Prices and Temperature



Results in 20-80% reduction in energy costs.

Smaller tests for algorithmic behavior

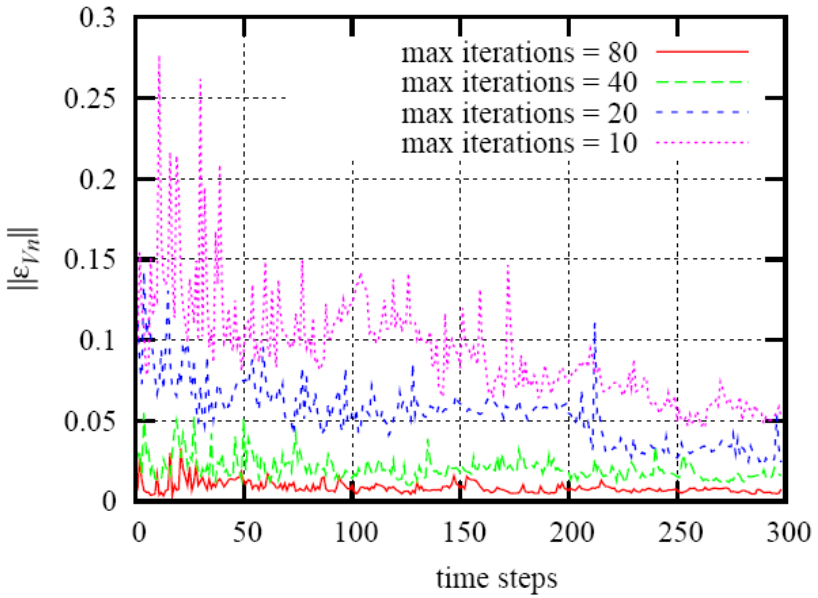
- Example: size-segregation in shaker, with thousands of steel spheres



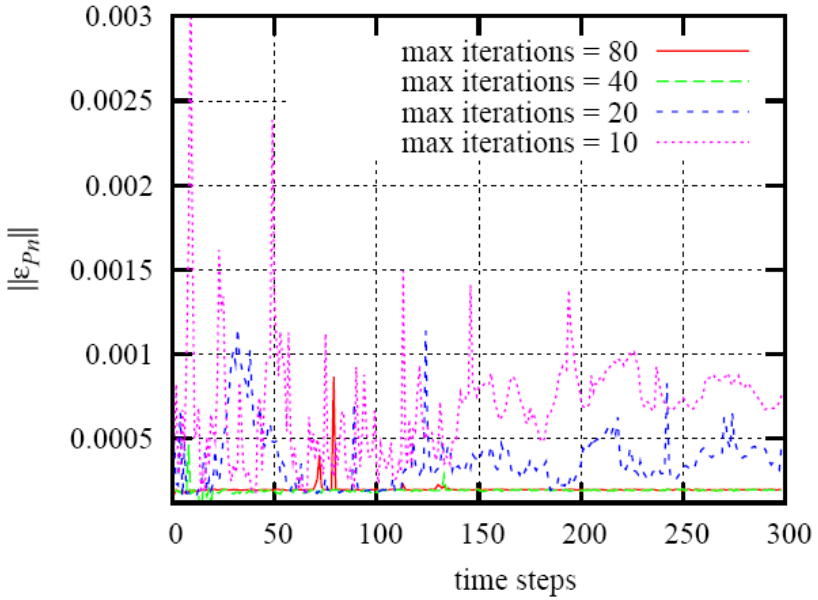
Note: solution beyond reach of Lemke-type LCP solvers!

Tests: Feasibilitiy

- Feasibility accuracy increases with number of iterations, method is consistent:



Speed violation in constraints

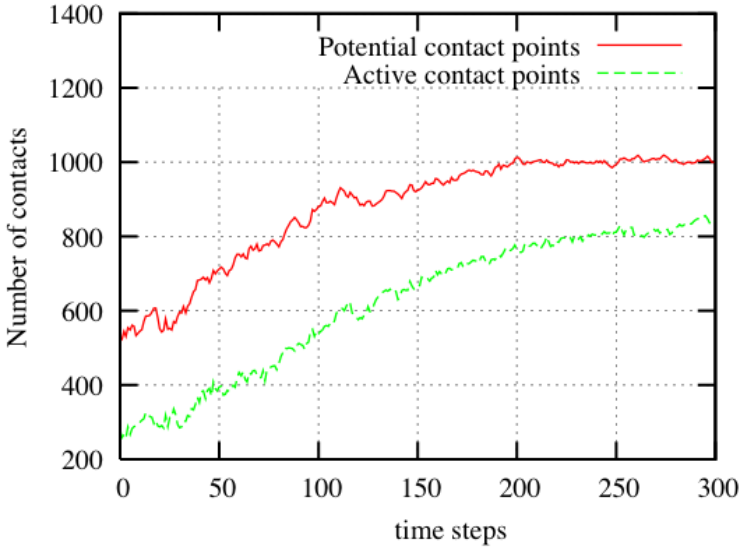


Position error in constraints (penetration)

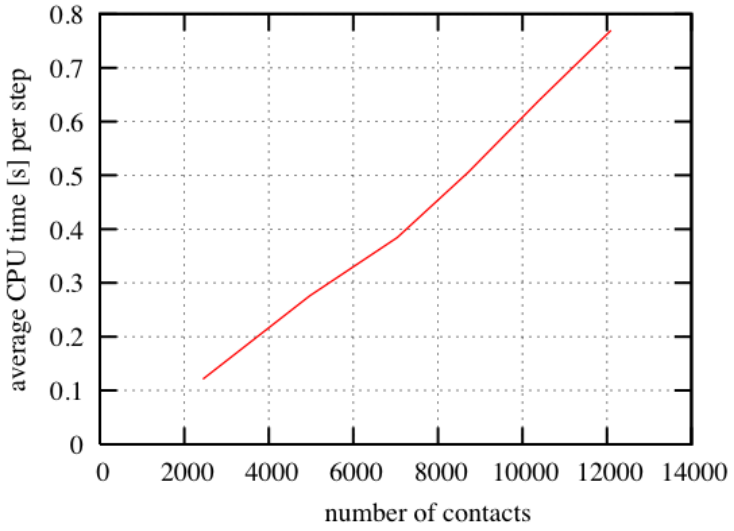
(with example of 300 spheres in shaker)

Tests: Scalability

- CPU effort per contact, since our contacts are the problem variables.
- Penetration error was uniformly no larger than 0.2% of diameter.



Number of contacts in time, 300 spheres



CPU time per step for 300-1500 spheres

General: Theory

$$(OC) \quad \begin{array}{ll} \min & f(x) = \frac{1}{2} x^T N x + r^T x \\ s.t. & x_i \in \Upsilon^i, \qquad\qquad\qquad i = 1, 2, \ldots, n_k. \end{array}$$

Theorem Assume that $x^0 \in \Upsilon$ and that the sequences of matrices B^r and K^r are bounded. Then we have that

$$f(x^{r+1}) - f(x^r) \leq -\beta \left\| x^{r+1} - x^r \right\|^2$$

for any iteration index r, and any accumulation point of the sequence x^r is a solution of (CCP).

Corollary Assume that the friction cone of the configuration is pointed The algorithm produces a bounded sequence, and any **accumulation point results in the same velocity solution**

We thus have an iterative and parallel-friendly algorithm.

DVI and Painleve paradoxes

- Unfortunately, there exist configurations for which no continuous solutions of the DVI will exist.

$$\begin{aligned} y' &= f(t, y(t), x(t)) \\ x(t) &\in SOL(K; F(t, y(t), \cdot)) \\ y(0) &= y_0 \end{aligned}$$

- Such configurations are called Painleve paradoxes, and appear only when friction is present. (Baraff 91, Stewart 98).
- We need weaker solution concepts. We use the one of measure differential inclusion (Stewart,98).

Differential Variational Inequalities– why do it?

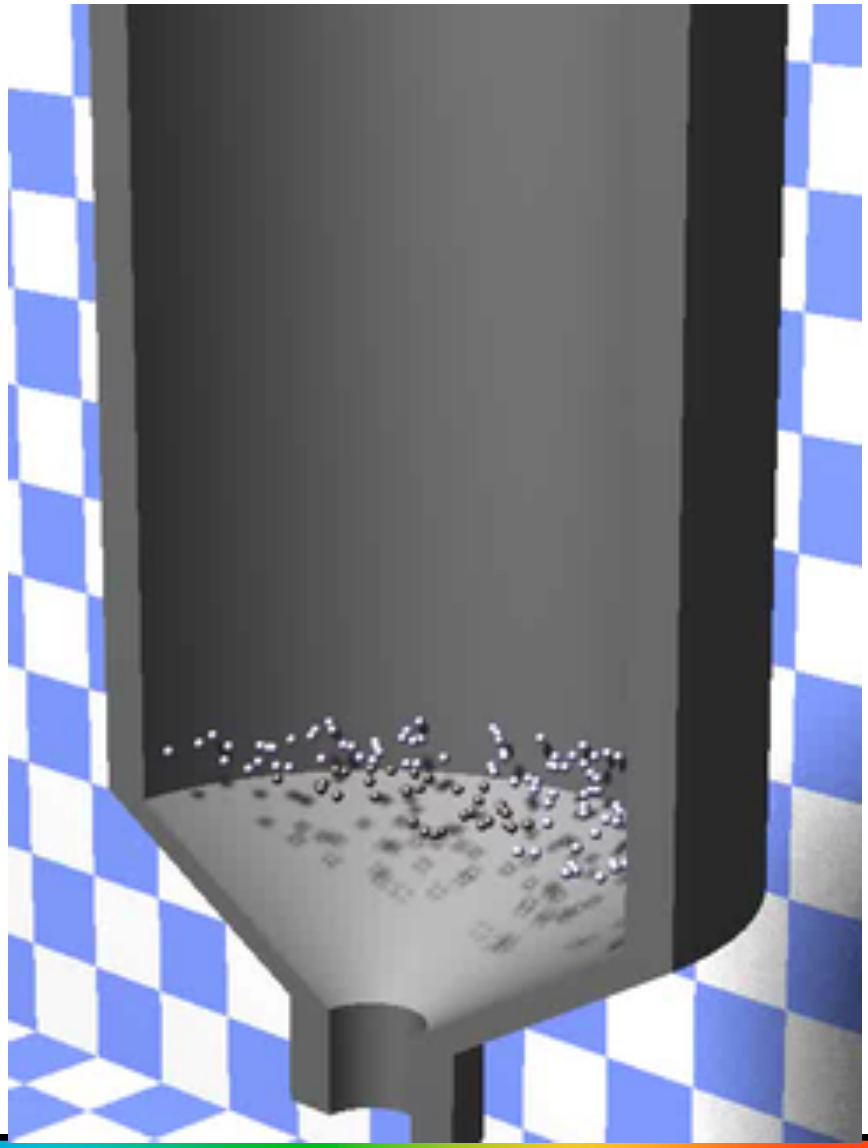
- Contact Dynamics.
  - Rigid-Bodies: Differential Operator is ODE.
  - Deformable Bodies: Differential Operator is PDE.
  - Granular Flow, Masonry Stability, Rock Dynamics…
- Finance: Option Pricing-- American Options. PDE-based.
- See Luo, Pang et al, and Kinderlehrer and Stampacchia Monographs..

It is a hybrid system – where is the switching?

- When bodies enter contact (collision, plastic in the previous formulation)
- Stick-Slip transition.

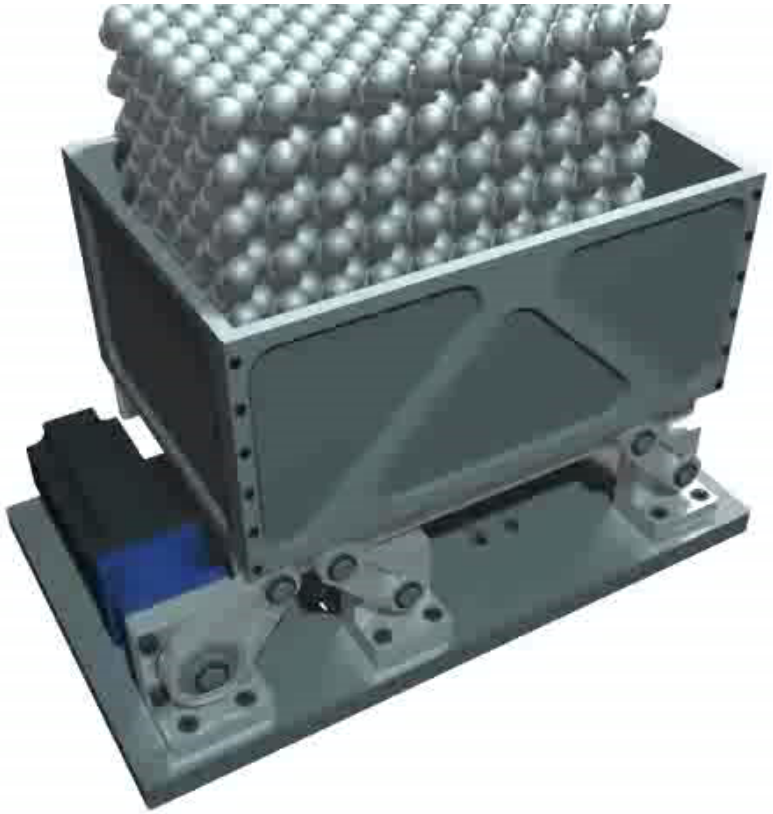
Simulating the PBR nuclear reactor

- 160’000 Uranium-Graphite spheres, 600’000 contacts on average
- Two millions of primal variables, six millions of dual variables
- *1 CPU day on a single processor…*
- We estimate 3CPU days, compare with 150 CPU days for DEM (Rycroft, Grest, et al.) !!!



Examples

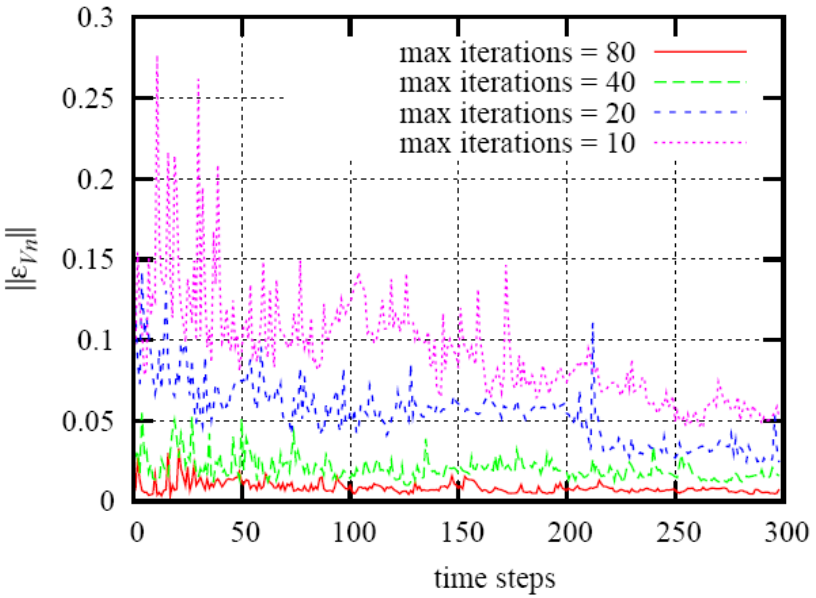
- Example: size-segregation in shaker, with thousands of steel spheres



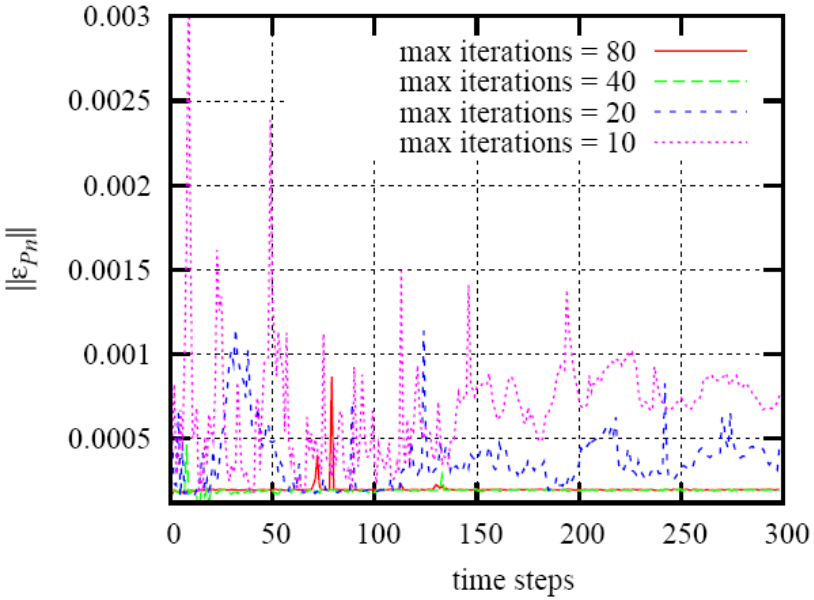
Note: solution beyond reach of Lemke-type LCP solvers!

Tests

- Feasibility accuracy increases with number of iterations:



Speed violation in constraints

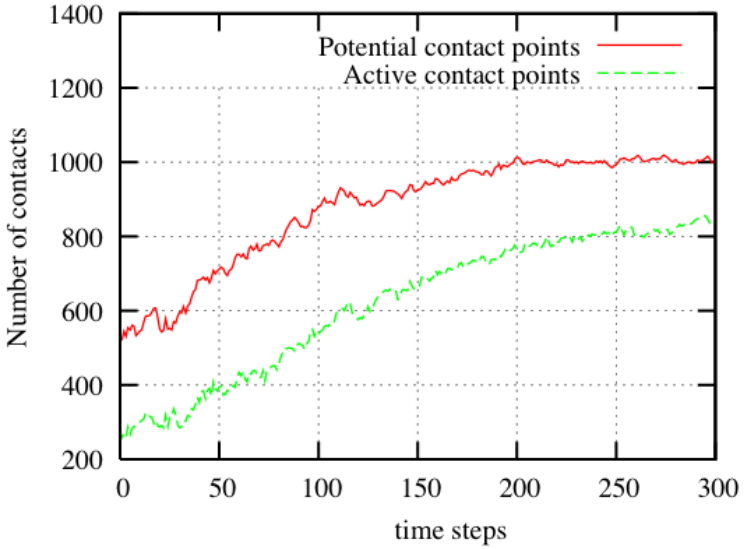


Position error in constraints (penetration)

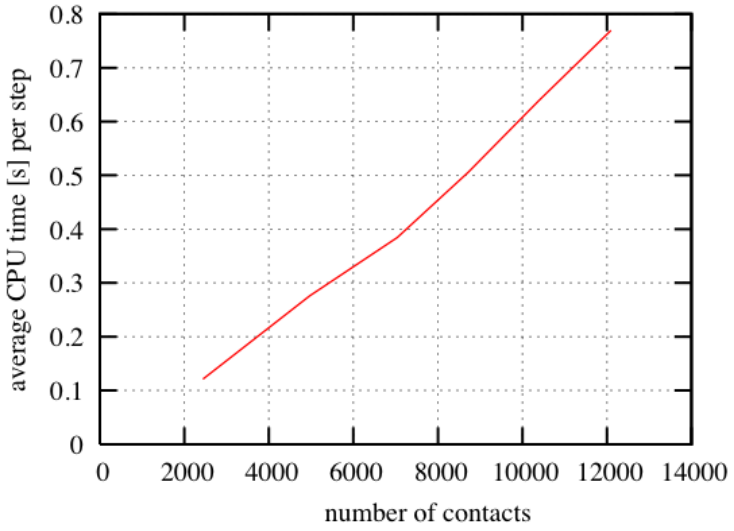
(with example of 300 spheres in shaker)

Tests: Scalability

- CPU effort per contact, since our contacts are the problem variables.
- Penetration error was uniformly no larger than 0.2% of diameter.



Number of contacts in time, 300 spheres

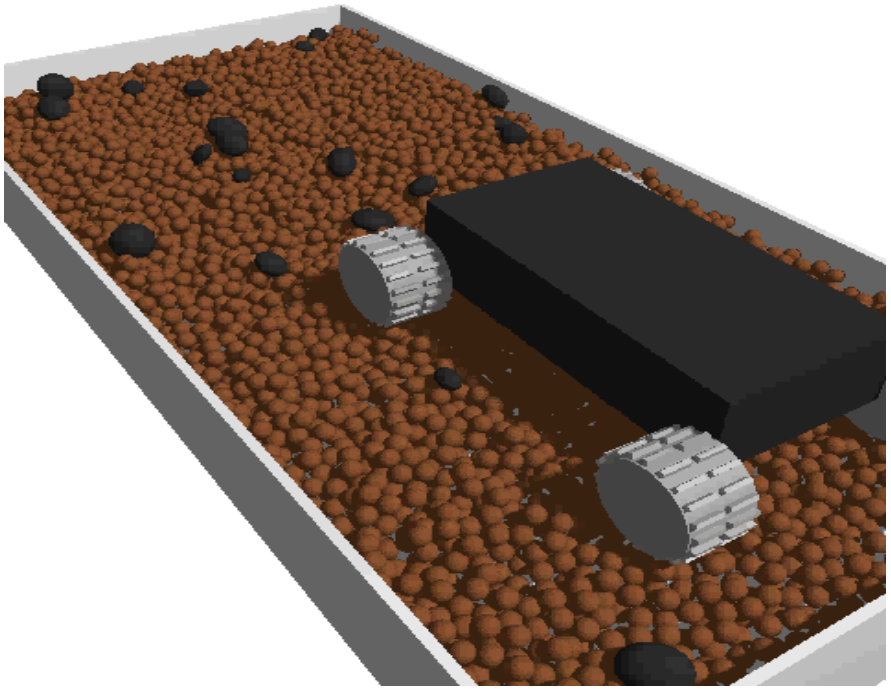


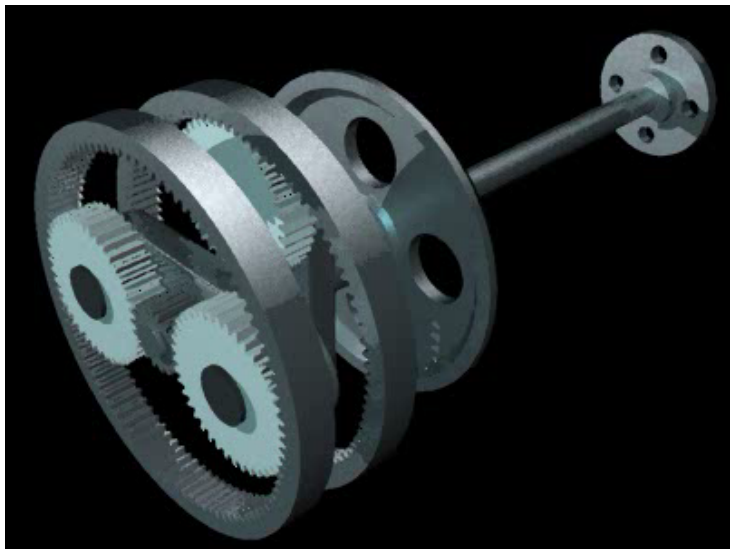
CPU time per step for 300-1500 spheres

IBM BlueGene/L—GPU comparison

- Entry model: 1024 dual core nodes
- 5.7 Tflop (compare to 0.5 Tflop for NVIDIA Tesla GPU)
- Dedicated OS
- Dedicated power management solution
- Require dedicated IT support
- Price (2007): $1.4 million
- Same GPU power (2008): 7K!!!

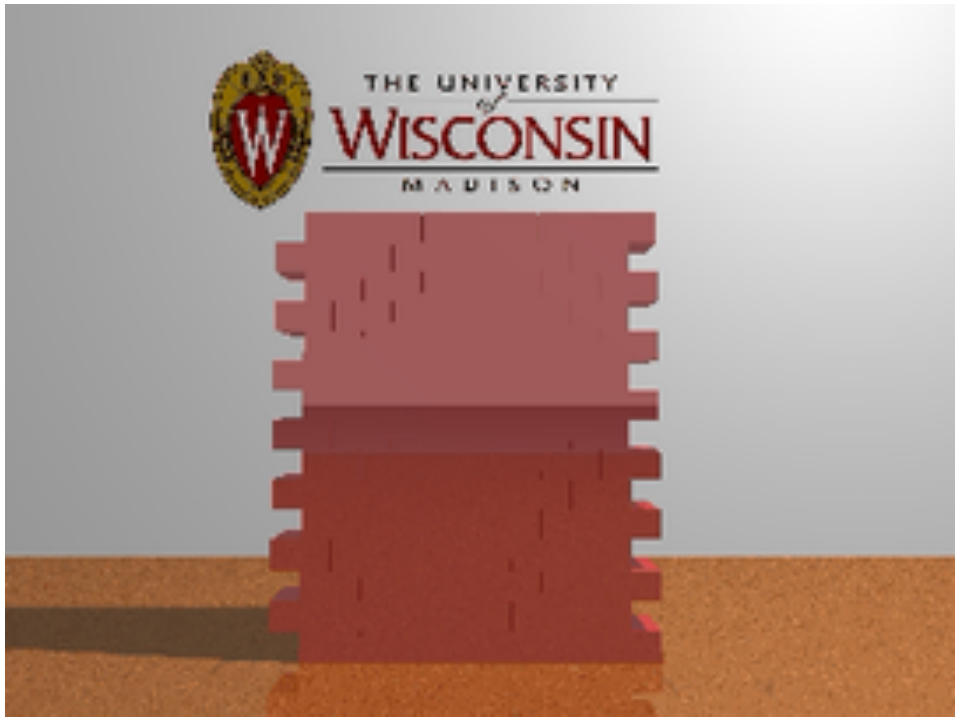
In addition, we can approach efficiently approach many engineering problems (see website for papers)





Brick Wall Example…

- Times reported are in seconds for one second long simulation
- GPU: NVIDIA GeForce 8800 GTX



Bricks	Sequential Version	GPU Co-processing Version
1000	43	6
2000	87	10
8000	319	42

Granular materials: applications

- Important? The second-most manipulated material in industry after water (Richard, Nature Materials 2005).
- Applications range from pharmaceutical, food, powders, petrochemical, nuclear, automotive, and semiconductor industries up to geological granular flows – some examples later.
- Two perhaps non-intuitive but crucial energy applications.
  - Circulating granular catalysts in refineries.
  - Fluidized bed coal gasification (“clean coal”).

Granular materials: Challenges, need for HPC

- The absence of a continuum theory makes particle-by-particle computational approaches the only general first principles computing approach – we need HPC.
- 1 m^3 of sand: ~1 trillion granules. Enormous … but just about within reach.
- In addition, the source of many open or difficult questions.
  - Is there a "random" close packing of spherical particles? (A maximum volume fraction of a random sphere population, Torquato et al., 2000). Postulated at ~0.636. Jamming.
  - Nothing is known of the same when there is friction.
  - The Kepler conjecture, proven in the last decade: in the deterministic case, the maximum space-filling volume fraction is the one of the cannonball arrangement: 0.7405

Are all interesting DVI problems over R+? No.

- Conic Complementarity IS NATURAL in granular dynamics (and MD).
- Coulomb model.

$$\left(\beta_1^{(j)}, \beta_2^{(j)}\right) = \operatorname{argmin}_{\mu^{(j)} c_n^{(j)} \geq \sqrt{\left(\beta_1^{(j)} + \beta_2^{(j)}\right)^2}} \left[\left(v^T t_1^{(j)}\right)\beta_1 + \left(v^T t_2^{(j)}\right)\beta_2\right]$$

$$K = \left\{(x,y,z) \middle| \mu^{(j)} z \geq \sqrt{y^2 + x^2}\right\} \quad K^* = \left\{(x,y,z) \middle| z \geq \mu^{(j)} \sqrt{y^2 + x^2}\right\}$$

$$\begin{pmatrix} c_n^{(j)} \\ \beta_1^{(j)} \\ \beta_2^{(j)} \end{pmatrix} \in K \quad \perp \quad \begin{pmatrix} \mu^{(j)} \sqrt{\left(v^T t_1^{(j)}\right)^2 + \left(v^T t_2^{(j)}\right)^2} \\ v^T t_1^{(j)} \\ v^T t_2^{(j)} \end{pmatrix} \in K^*$$

- Most previous time-stepping discretize friction cone to use LCP…
- Can we accommodate non-R+ cones naturally?

Granular materials: abstraction: DVI

- Differential variational inequalities (DVI, Stewart and Pang, 03-08): Mixture of differential equations and variational inequalities.

$$\begin{aligned} y' &= f(t, y(t), x(t)) \\ x(t) &\in SOL(K; F(t, y(t), \cdot)) \\ y(0) &= y_0 \end{aligned}$$

$$x \in SOL(K; F(t, y, \cdot) \Leftrightarrow (\tilde{x} - x)^T F(t, y, x) \geq 0, \forall \tilde{x} \in K$$

- In the case of complementarity, $K = \mathbb{R}^n_+$

$$\begin{aligned} y' &= f(t, y(t), x(t)) \\ 0 &\leq x(t); F(t, y(t), x(t)) \\ 0 &= x(t)^T F(t, y(t), x(t)) \\ y(0) &= y_0 \end{aligned}$$

- Many, but not all of our conclusions will be extensible or applicable to this form of DVI,

Granular materials: abstraction: DVI

- Differential variational inequalities (DVI, Stewart and Pang, 03-08): Mixture of differential equations and variational inequalities.

$$\begin{aligned} y' &= f(t, y(t), x(t)) \\ x(t) &\in SOL(K; F(t, y(t), \cdot)) \\ y(0) &= y_0 \end{aligned}$$

$$x \in SOL(K; F(t, y, \cdot) \Leftrightarrow (\tilde{x} - x)^T F(t, y, x) \geq 0, \forall \tilde{x} \in K$$

- In the case of complementarity, $K = \mathbb{R}^n_+$

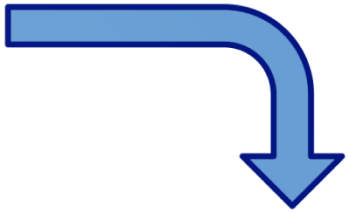
$$\begin{aligned} y' &= f(t, y(t), x(t)) \\ 0 &\leq x(t); F(t, y(t), x(t)) \\ 0 &= x(t)^T F(t, y(t), x(t)) \\ y(0) &= y_0 \end{aligned}$$

- Many, but not all of our conclusions will be extensible or applicable to this form of DVI,

DVI: time-stepping methods

- Our target methodology are time stepping methods.

$$
\begin{aligned}
y' &= f(t, y(t), x(t)) \\
x(t) &\in SOL(K; F(t, y(t), \cdot)) \\
y(0) &= y_0
\end{aligned}
$$



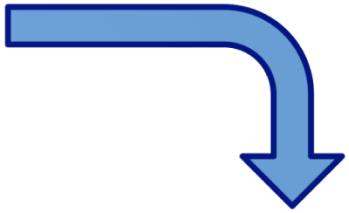
$$
\begin{aligned}
y^{h,(i+1)} &= y^{h,i} + h\tilde{f}\left(\tilde{t}^{h,(i+1)}, \theta_1 y^{h,i} + (1-\theta_1) y^{h,(i+1)}, x^{h,(i+1)}\right) \\
x^{h,(i+1)} &\in SOL(K; \tilde{F}(\tilde{t}^{h,(i+1)}, \theta_2 y^{h,i} + (1-\theta_2) y^{h,(i+1)}, \cdot)) \\
y(0) &= y_0.
\end{aligned}
$$

- Implicit in the VI variable, implicit-explicit in state, with possible linearization of the structural functions f,F.
- Promises much larger time steps than smoothing with explicit integration: key to its stability and superior performance.

DVI: time-stepping methods

- Our target methodology are time stepping methods.

$$
\begin{aligned}
y' &= f(t, y(t), x(t)) \\
x(t) &\in SOL(K; F(t, y(t), \cdot)) \\
y(0) &= y_0
\end{aligned}
$$

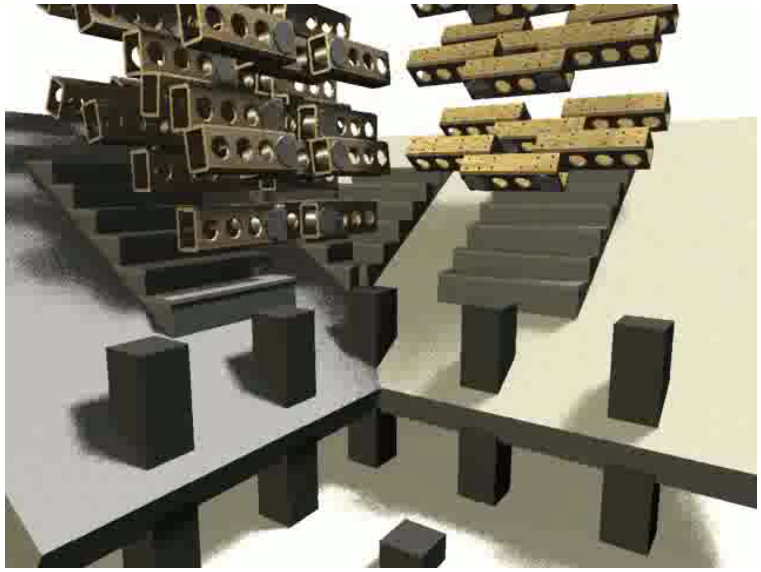


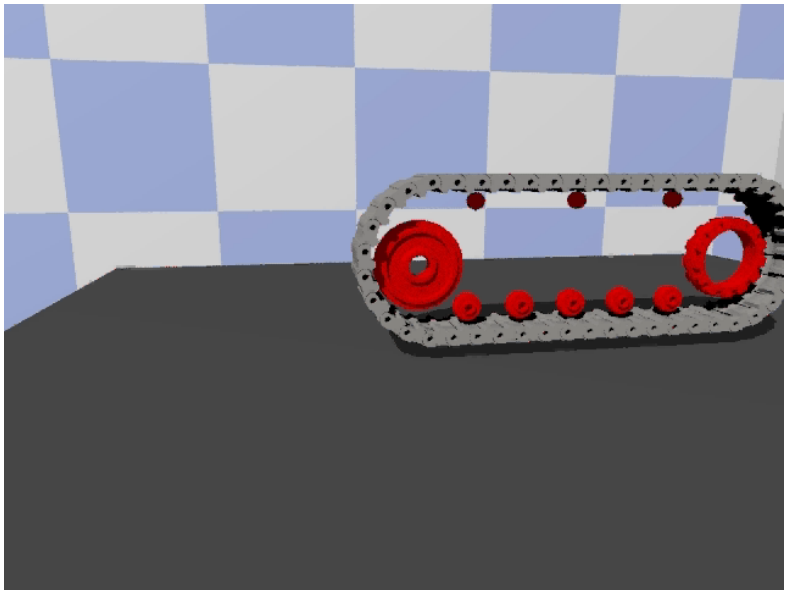
$$
\begin{aligned}
y^{h,(i+1)} &= y^{h,i} + h\tilde{f}\left(\tilde{t}^{h,(i+1)}, \theta_1 y^{h,i} + (1-\theta_1) y^{h,(i+1)}, x^{h,(i+1)}\right) \\
x^{h,(i+1)} &\in SOL(K; \tilde{F}(\tilde{t}^{h,(i+1)}, \theta_2 y^{h,i} + (1-\theta_2) y^{h,(i+1)}, \cdot)) \\
y(0) &= y_0.
\end{aligned}
$$

- Implicit in the VI variable, implicit-explicit in state, with possible linearization of the structural functions f,F.
- Promises much larger time steps than smoothing with explicit integration: key to its stability and superior performance.

Other applications of DVI:

- Physics-based virtual reality.
- Automotive design
- Dynamics of multicristalline materials: evolution of the boundary between phases.
- Porous Media Flow.
- …..
- Generally appears any time dynamics and “switching” is encountered.





Are all interesting DVI problems over R+? No.

- Conic Complementarity IS NATURAL in granular dynamics (and MD).
- Coulomb model.

$$\left(\beta_1^{(j)}, \beta_2^{(j)}\right) = \operatorname{argmin}_{\mu^{(j)} c_n^{(j)} \geq \sqrt{\left(\beta_1^{(j)} + \beta_2^{(j)}\right)^2}} \left[\left(v^T t_1^{(j)}\right)\beta_1 + \left(v^T t_2^{(j)}\right)\beta_2\right]$$

$$K = \left\{(x,y,z) \middle| \mu^{(j)} z \geq \sqrt{y^2 + x^2}\right\} \quad K^* = \left\{(x,y,z) \middle| z \geq \mu^{(j)} \sqrt{y^2 + x^2}\right\}$$

$$\begin{pmatrix} c_n^{(j)} \\ \beta_1^{(j)} \\ \beta_2^{(j)} \end{pmatrix} \in K \quad \perp \quad \begin{pmatrix} \mu^{(j)} \sqrt{\left(v^T t_1^{(j)}\right)^2 + \left(v^T t_2^{(j)}\right)^2} \\ v^T t_1^{(j)} \\ v^T t_2^{(j)} \end{pmatrix} \in K^*$$

- Most previous time-stepping discretize friction cone to use LCP…
- Can we accommodate non-R+ cones naturally?

Granular materials: abstraction: DVI

- Differential variational inequalities (DVI, Stewart and Pang, 03-08): Mixture of differential equations and variational inequalities.

$$\begin{aligned} y' &= f(t, y(t), x(t)) \\ x(t) &\in SOL(K; F(t, y(t), \cdot)) \\ y(0) &= y_0 \end{aligned}$$

$$x \in SOL(K; F(t, y, \cdot) \Leftrightarrow (\tilde{x} - x)^T F(t, y, x) \geq 0, \forall \tilde{x} \in K$$

- In the case of complementarity, $K = \mathbb{R}^n_+$

$$\begin{aligned} y' &= f(t, y(t), x(t)) \\ 0 &\leq x(t); F(t, y(t), x(t)) \\ 0 &= x(t)^T F(t, y(t), x(t)) \\ y(0) &= y_0 \end{aligned}$$

- Many, but not all of our conclusions will be extensible or applicable to this form of DVI,

Granular materials: abstraction: DVI

- Differential variational inequalities (DVI, Stewart and Pang, 03-08): Mixture of differential equations and variational inequalities.

$$\begin{aligned} y' &= f(t, y(t), x(t)) \\ x(t) &\in SOL(K; F(t, y(t), \cdot)) \\ y(0) &= y_0 \end{aligned}$$

$$x \in SOL(K; F(t, y, \cdot) \Leftrightarrow (\tilde{x} - x)^T F(t, y, x) \geq 0, \forall \tilde{x} \in K$$

- In the case of complementarity, $K = \mathbb{R}^n_+$

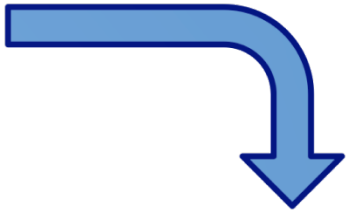
$$\begin{aligned} y' &= f(t, y(t), x(t)) \\ 0 &\leq x(t); F(t, y(t), x(t)) \\ 0 &= x(t)^T F(t, y(t), x(t)) \\ y(0) &= y_0 \end{aligned}$$

- Many, but not all of our conclusions will be extensible or applicable to this form of DVI,

DVI: time-stepping methods

- Our target methodology are time stepping methods.

$$
\begin{aligned}
y' &= f(t, y(t), x(t)) \\
x(t) &\in SOL(K; F(t, y(t), \cdot)) \\
y(0) &= y_0
\end{aligned}
$$



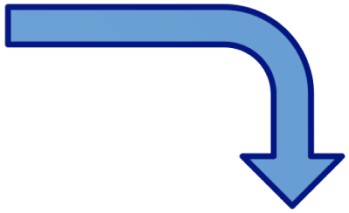
$$
\begin{aligned}
y^{h,(i+1)} &= y^{h,i} + h\tilde{f}\left(\tilde{t}^{h,(i+1)}, \theta_1 y^{h,i} + (1-\theta_1) y^{h,(i+1)}, x^{h,(i+1)}\right) \\
x^{h,(i+1)} &\in SOL(K; \tilde{F}(\tilde{t}^{h,(i+1)}, \theta_2 y^{h,i} + (1-\theta_2) y^{h,(i+1)}, \cdot)) \\
y(0) &= y_0.
\end{aligned}
$$

- Implicit in the VI variable, implicit-explicit in state, with possible linearization of the structural functions f,F.
- Promises much larger time steps than smoothing with explicit integration: key to its stability and superior performance.

DVI: time-stepping methods

- Our target methodology are time stepping methods.

$$
\begin{aligned}
y' &= f(t, y(t), x(t)) \\
x(t) &\in SOL(K; F(t, y(t), \cdot)) \\
y(0) &= y_0
\end{aligned}
$$

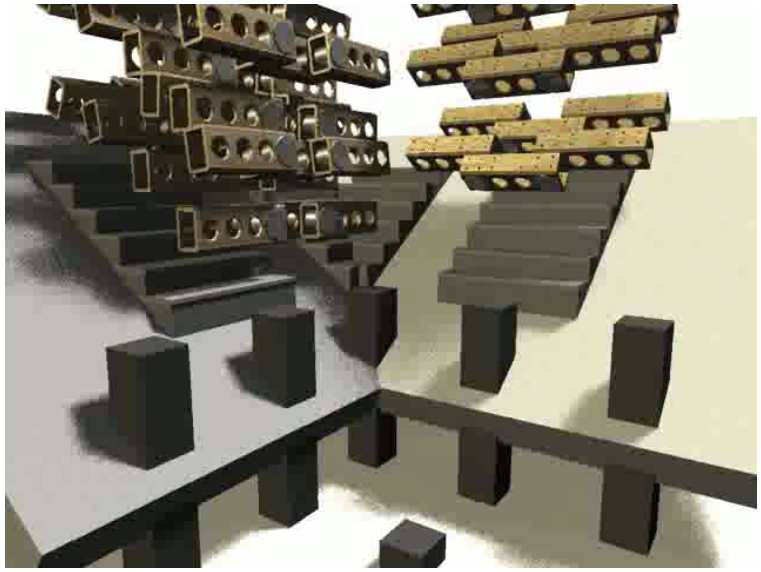


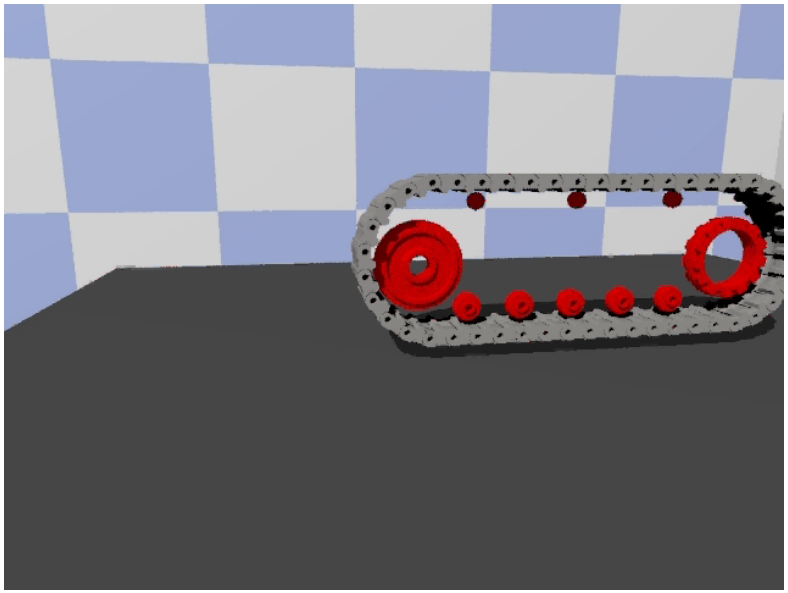
$$
\begin{aligned}
y^{h,(i+1)} &= y^{h,i} + h\tilde{f}\left(\tilde{t}^{h,(i+1)}, \theta_1 y^{h,i} + (1-\theta_1) y^{h,(i+1)}, x^{h,(i+1)}\right) \\
x^{h,(i+1)} &\in SOL(K; \tilde{F}(\tilde{t}^{h,(i+1)}, \theta_2 y^{h,i} + (1-\theta_2) y^{h,(i+1)}, \cdot)) \\
y(0) &= y_0.
\end{aligned}
$$

- Implicit in the VI variable, implicit-explicit in state, with possible linearization of the structural functions f,F.
- Promises much larger time steps than smoothing with explicit integration: key to its stability and superior performance.

Other applications of DVI:

- Physics-based virtual reality.
- Automotive design
- Dynamics of multicristalline materials: evolution of the boundary between phases.
- Porous Media Flow.
- …..
- Generally appears any time dynamics and “switching” is encountered.





Content: The road to …

- Solving DVI
  - Smoothing versus hard constraints
  - (1) Time Stepping
  - Nonconvexity – (2) Convex Approximation:
  - Some Theory
- (3) Iterative Algorithms for the time-stepping subproblem.
- Results
  - Validation.
  - GPU implementation
  - Application Examples
- Open problems/extensions.

Time-stepping scheme

- Write an implicit-explicit scheme AS IF Painleve paradoxes do not exist.
- We use linearization (Anitescu and Hart, 04) NOT index reduction : results in constraint stabilization.
- Proceed with a fixed time step. Collisions are forced to be “simultaneous” within one step.

Further insight.

- The key is the combination between relaxation and constraint stabilization.

$$0 \le \frac{1}{h}\Phi^{(j)}\left(q^{(l)}\right)+\nabla_q \Phi^{(j)}\left(q^{(l)}\right)v^{(l+1)} - \mu^{(j)}\sqrt{\left(D_u^{l,t}v\right)^2+\left(D_v^{l,t}v\right)^2}$$

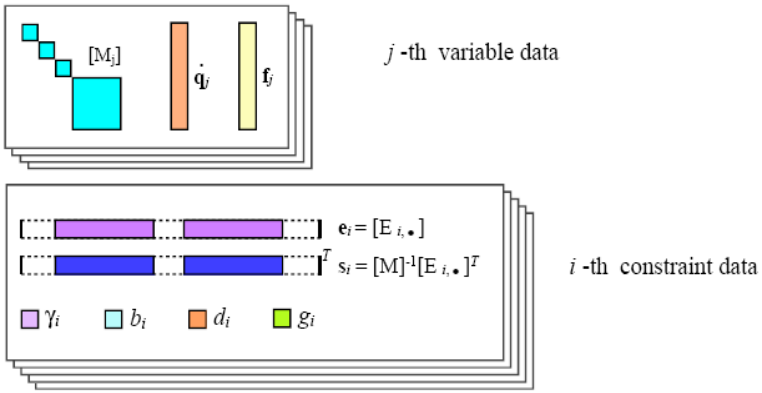
- If the time step is smaller than the variation in velocity then the gap function settles at

$$0 \approx \frac{1}{h}\Phi^{(j)}\left(q^{(l)}\right)- \mu^{(j)}\sqrt{\left(D_u^{l,t}v\right)^2+\left(D_v^{l,t}v\right)^2}$$

- So the solution is the same as the original scheme for a slightly perturbed gap function.

The algorithm

- Development of an efficient algorithm for fixed point iteration:



- *avoid temporary data, exploit sparsity. Never compute explicitly the N matrix!*
- *implemented in incremental form. Compute only deltas of multipliers.*
- *O(n) space requirements and supports premature termination*

The algorithm is specialized, for minimum memory use!

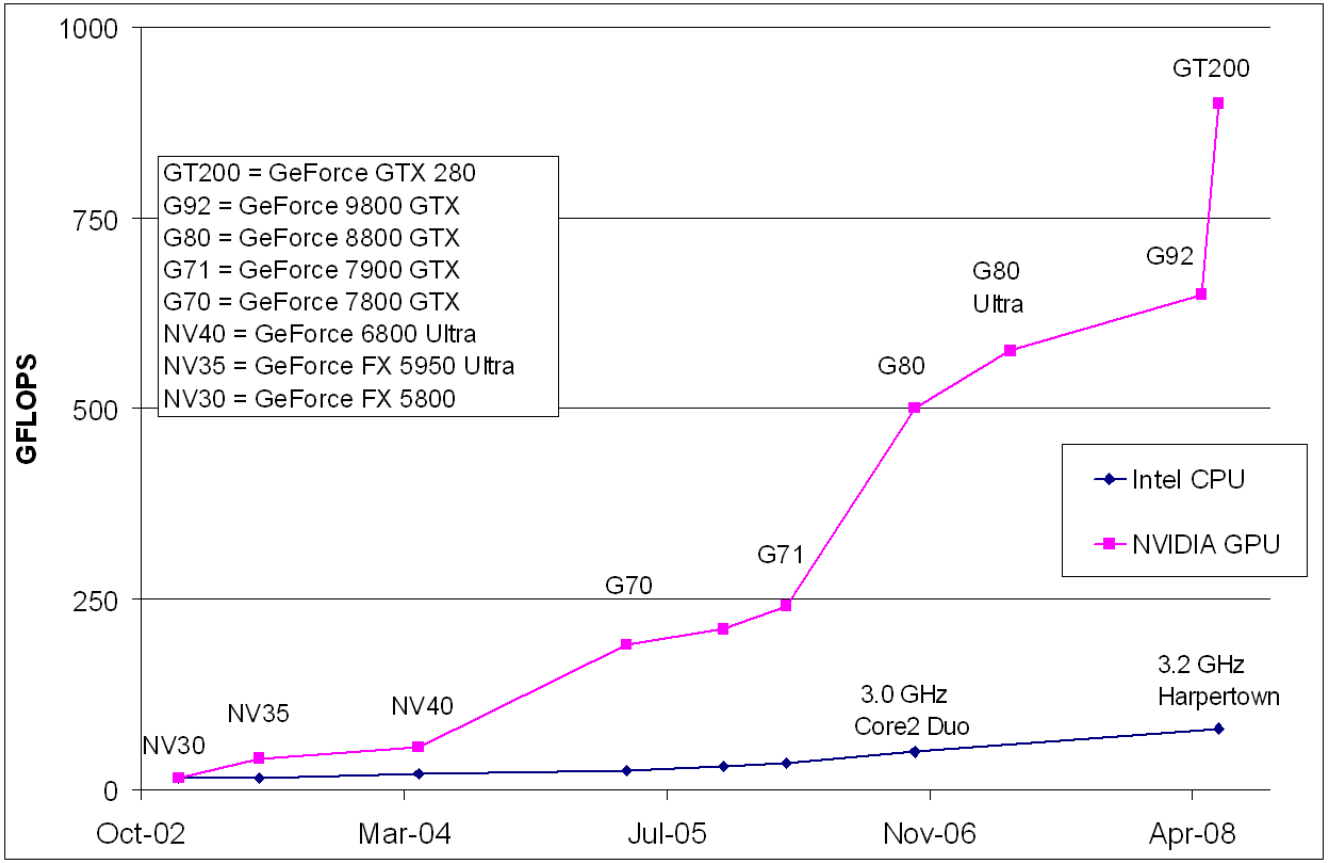
COM: Constraint <-> Body

(1) // Pre-compute some data for friction constraints
(2) for $i := 1$ to $n_{\mathcal{A}}$
(3) $\quad \boldsymbol{s}_a^i = M^{-1} D^i$
(4) $\quad g_a^i = D^{i,T} \boldsymbol{s}_a^i$
(5) $\quad \eta_a^i = \frac{3}{\mathrm{Trace}(g_a^i)}$
(6) // Pre-compute some data for bilateral constraints
(7) for $i := 1$ to $n_{\mathcal{B}}$
(8) $\quad s_b^i = M^{-1} \nabla \Psi^i$
(9) $\quad g_b^i = \nabla \Psi^{i,T} s_b^i$
(10) $\quad \eta_b^i = \frac{1}{g_b^i}$
(11)
(12) // Initialize impulses
(13) if warm start with initial guess $\boldsymbol{\gamma}_{\mathcal{E}}^*$
(14) $\quad \boldsymbol{\gamma}_{\mathcal{E}}^0 = \boldsymbol{\gamma}_{\mathcal{E}}^*$
(15) else
(16) $\quad \boldsymbol{\gamma}_{\mathcal{E}}^0 = 0$
(17)
(18) // Initialize speeds
(19) $\boldsymbol{v} = \sum_{i=1}^{n_{\mathcal{A}}} \boldsymbol{s}_a^i \boldsymbol{\gamma}_a^{i,0} + \sum_{i=1}^{n_{\mathcal{B}}} s_b^i \gamma_b^{i,0} + M^{-1} \tilde{\boldsymbol{k}}$

(21) // Main iteration loop
(22) for $r := 0$ to r_{max}
(23) $\quad$ // Loop on frictional constraints
(24) $\quad$ for $i := 1$ to $n_{\mathcal{A}}$
(25) $\quad\quad \boldsymbol{\delta}_a^{i,r} = \left(\boldsymbol{\gamma}_a^{i,r} - \omega \eta_a^i \left(D^{i,T} \boldsymbol{v}^r + \boldsymbol{b}_a^i\right)\right)$;
(26) $\quad\quad \boldsymbol{\gamma}_a^{i,r+1} = \lambda \Pi_{\Upsilon}\left(\boldsymbol{\delta}_a^{i,r}\right) + (1-\lambda)\boldsymbol{\gamma}_a^{i,r}$;
(27) $\quad\quad \Delta \boldsymbol{\gamma}_a^{i,r+1} = \boldsymbol{\gamma}_a^{i,r+1} - \boldsymbol{\gamma}_a^{i,r}$;
(28) $\quad\quad \boldsymbol{v} := \boldsymbol{v} + \boldsymbol{s}_a^{i\,T} \Delta \boldsymbol{\gamma}_a^{i,r+1}$.
(29) $\quad$ // Loop on bilateral constraints
(30) $\quad$ for $i := 1$ to $n_{\mathcal{B}}$
(31) $\quad\quad \delta_b^{i,r} = \left(\gamma_b^{i,r} - \omega \eta_b^i \left(\nabla \Psi^{i,T} \boldsymbol{v}^r + b_b^i\right)\right)$;
(32) $\quad\quad \gamma_b^{i,r+1} = \lambda \Pi_{\Upsilon}\left(\delta_b^{i,r}\right) + (1-\lambda)\gamma_b^{i,r}$;
(33) $\quad\quad \Delta \gamma_b^{i,r+1} = \gamma_b^{i,r+1} - \gamma_b^{i,r}$;
(34) $\quad\quad \boldsymbol{v} := \boldsymbol{v} + s_b^{i\,T} \Delta \gamma_b^{i,r+1}$.
(35)
(36) return $\boldsymbol{\gamma}_{\mathcal{E}}$, $\boldsymbol{v}$

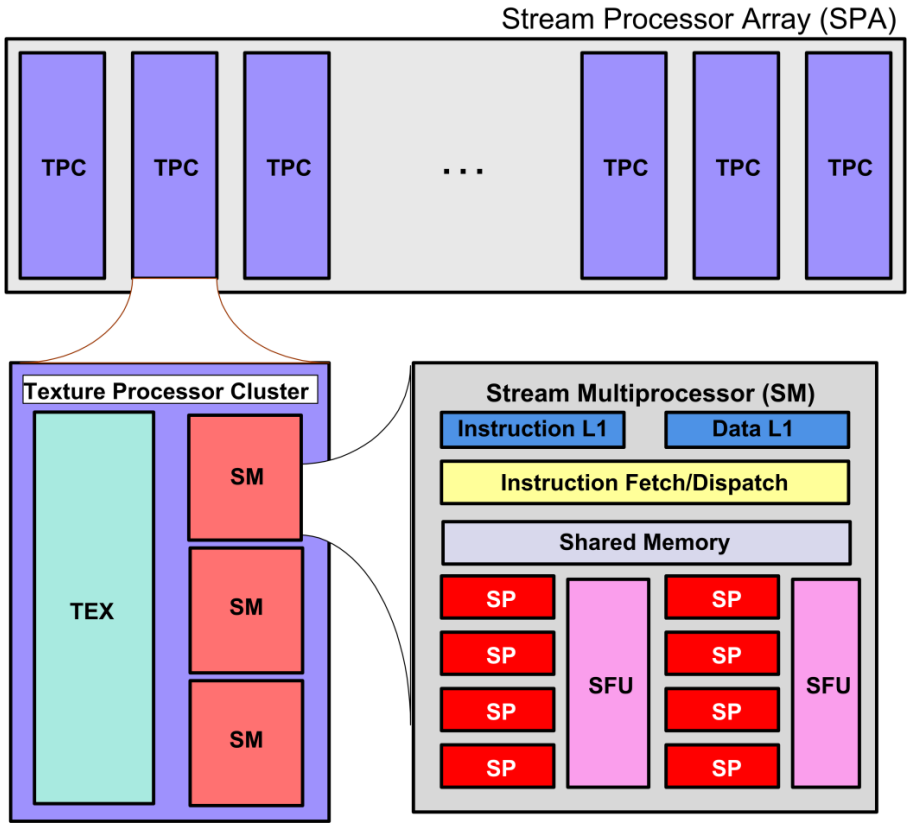
The rest “in place” per-body or per-constraint for Gauss-Jacobi

GPU : The attraction.



- Your PC graphic board is a supercomputer (0.32TF, GT8800).
- 5.7 TF: IBM BG/L $1,400K (2007) – NVIDIA Tesla $7K (2008)

NVIDIA TESLA C1060



- 30 Stream Multiprocessors.
- 240 Scalar Processors
- 4 GB device memory
- Memory Bandwidth: 102 GB/s
- Clock Rate: 1.3GHz
- Approx. $1,250

Parallel CCP on GPU: The 30,000 Feet Perspective

- Relies on a Gauss-Jacobi iteration: the first step.
- The GPU is viewed as a compute device that:
  - Is a co-processor to the CPU or host
  - Has its own DRAM (device memory)
  - Runs many threads in parallel (30K)
- Data-parallel portions (such as per-body "in-place") of an application are executed on the device as kernels which run in parallel on many threads
- Each simulation time step invokes multiple GPU calls
  - For each of these calls, parallelism can be on a
    - *"Per body" basis (work is done on different bodies in parallel)*
    - *"Per contact" basis (different contact events are processed in parallel)*

GPU: The CCP Pre-Processing

1. (*GPU, body-parallel*) **Force kernel**. For each body, compute applied external forces $\mathbf{f}(t^{(l)}, \mathbf{q}^{(l)}, \mathbf{v}^{(l)})$ (for example, gravitational and gyroscopic forces). Produce the force $\mathbf{F}_j$ and the torque $\mathbf{C}_j$ acting at CM of each body j.

2. (*GPU, contact-parallel*) **Contact preprocessing kernel**. For each contact i, given contact normal and position, compute in-place the matrices $\mathbf{D}^T_{i,v_A}$, $\mathbf{D}^T_{i,\omega_A}$ and $\mathbf{D}^T_{i,\omega_B}$, and the contact residual $\mathbf{b}_i = \{\frac{1}{h}\Phi_i(\mathbf{q}), 0, 0\}^T$.

3. (*GPU, body-parallel*) **Velocity Initialization kernel**. For each body j, initialize body velocity corrections: $\Delta\dot{\mathbf{r}}_j^{(l+1)} = h\ m_j^{-1}\mathbf{F}_j$ and $\Delta\omega_j^{(l+1)} = h\ \mathbf{J}_j^{-1}\mathbf{C}_j$.

GPU: The CCP Loop

4. (*GPU, contact-parallel*) **CCP iteration kernel**. For each contact i, do $\gamma_i^{r+1} = \lambda\ \Pi_{\Upsilon_i}\left(\gamma_i^r - \omega\eta_i\left(\mathbf{D}_i^T\mathbf{v}^r + \mathbf{b}_i\right)\right) + (1-\lambda)\gamma_i^r$. Store $\Delta\gamma_i^{r+1} = \gamma_i^{r+1} - \gamma_i^r$ in contact buffer. Compute updates to the velocities of the two connected bodies A and B (like $\Delta\dot{\mathbf{r}}_{A_i}^{(l+1)} = m_{A_i}^{-1}\mathbf{D}_{i,v_A}\Delta\gamma_i^{r+1}$, $\Delta\omega_{A_i}^{(l+1)} = \mathbf{J}_{A_i}^{-1}\mathbf{D}_{i,\omega_A}\Delta\gamma_i^{r+1}$), and store them in the reduction buffer.

5. (*GPU, reduction-slot-parallel*) **Run body-velocity reduction kernel**.

6. (*GPU, body-parallel*) **Body velocity updates kernel**. For each j body, add the cumulative velocity updates: $\dot{\mathbf{r}}_j^{(l+1)} = \dot{\mathbf{r}}_j^{(l)} + \Delta\dot{\mathbf{r}}_j^{(l+1)}$, and $\omega_j^{(l+1)} = \omega_j^{(l)} + \Delta\omega_j^{(l+1)}$.

7. Repeat from step 4 until convergence or until number of CCP iterations reached $r > r_{max}$.

8. (*GPU, body-parallel*) **Time integration kernel**. For each j body, perform time integration as $\mathbf{q}_j^{(l+1)} = \mathbf{q}_j^{(l)} + h\mathbf{L}(\mathbf{q}_j^{(l)})\mathbf{v}_j^{(l+1)}$

9. (*CPU, serial*) If post processing required, fetch body data structures and contact multipliers from GPU memory to host memory.

Collision detection on the GPU

- For granular dynamics, the number of force multipliers (x) is, in principle proportional to *the square* of the number of bodies. 1 trillion bodies -> 10^24 multipliers.

$$\begin{aligned} y' &= f(t, y(t), x(t)) \\ x(t) &\in SOL(K; F(t, y(t), \cdot)) \\ y(0) &= y_0 \end{aligned}$$

- Collision detection is used to reduce the active set to the one of the multipliers of pairs of bodies that *could be* in contact – *brute force still N^2*.
- A binning strategy is used to reduce the complexity (Negrut et al. 2009)..

Scalable 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*

Key Components, CD Method

- The method proposed draws on
  - Sorting (Radix Sort)
    - *O(N) parallel implementation*
  - Exclusive Prefix Scan
    - *O(N) parallel implementation*
  - Fast binning operation for the simple convex geometries
    - *On a rectangular grid it is very easy to figure out where the center of a sphere lands*