Solving Large-Scale Differential Variational Inequalities in Dense Granular Flow

Mihai ANITESCU ,

Argonne National Laboratory

With Dan Negrut,(Wisconsin) and A. Tasora (Parma)

OPTIM-A UIUC 2009

Argonne NATIONAL LABORATORY

... for a brighter future

U.S. Department of Energy

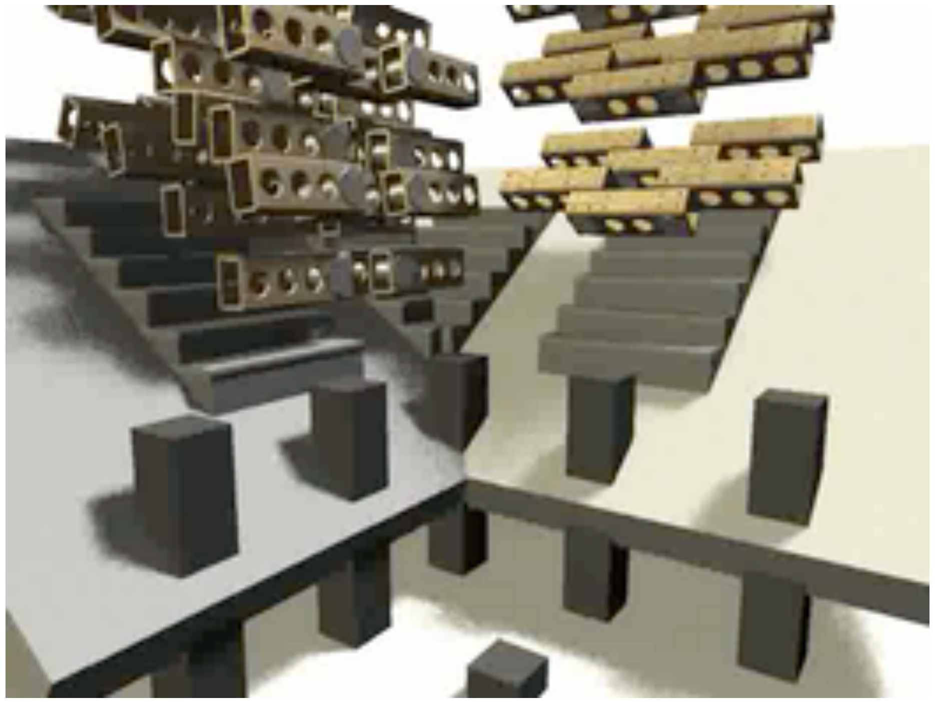
THE UNIVERSITY OF CHICAGO

Office of Science U.S. DEPARTMENT OF ENERGY

A U.S. Department of Energy laboratory managed by The University of Chicago

Contact dynamics applications

- Granular Flow,
- Masonry Stability,
- Rock Dynamics.
- Agent-Based Modeling (Pedestrian Evacuation Dynamics).
- Physics-based graphics simulations.
- Most common approaches by far are smoothing approaches (DEM)



What is DVI?

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

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

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

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

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

DVI: This session.

- Mihai Anitescu: Large-scale DVI appearing in Granular flow.
- Vijay Kumar: DVI for robotics manipulation with cables.
- Dan Negrut: Solving large-scale DVI on GPU.

Nonsmooth contact dynamics—a type of DVI

- Differential problem with variational inequality constraints – DVI

Newton Equations

Non-Penetration Constraints

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

Generalized Velocities

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

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

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

Friction Model

- Truly, a Differential Problem with Equilibrium Constraints

DVIs: Further consideration

- DVIs recently identified as distinguished problem class mixing ODE and VI by Pang and Stewart.
- DVIs appear whenever both dynamics and inequalities/ switching appear in model description.
- Dense granular flow. The second most-manipulated industrial material after water!
  - It still has no suitable continuum theory, since it can exhibit simultaneous gas, liquid or solid behavior.
  - Direct numerical simulation—like ours -- and experiments are the only discovery tools.
  - It appears in PBR, and pebble bed breeding blanket in ITER.

DVI: Time-stepping

- Idea: Approximate with a discrete time dynamical system which is stable and enforces inequality constraints exactly.
- Generic time-stepping scheme (Pang and Stewart)

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

- Investigations in this area have build on work by Moreau (196*), Glocker and Pfeiffer, Stewart and Trinkle, Pang and Trinkle, and many others.

Smoothing/regularization/DEM

- Recall, DVI (for C=R+) $\longrightarrow$

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

- Smoothing $\longrightarrow$

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

- Followed by forward Euler. Easy to implement!! $\longrightarrow$

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

- Compare with the complexity of time-stepping $\longrightarrow$

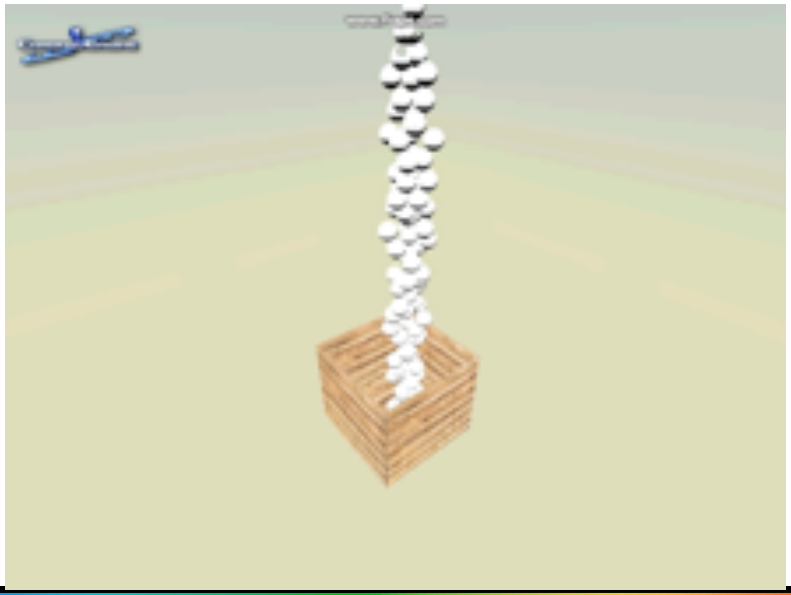
$$x^{n+1} = x^n + hf\left(t^{n+1}, x^{n+1}, u^{n+1}\right);$$
$$u^{n+1} \geq 0 \perp F\left(t^{n+1}, x^{n+1}, u^{n+1}\right) \geq 0$$

- Which is faster?

Smoothing (1) 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

Last_Run Time= 0.0000 Frame=001



ADAMS versus ChronoEngine

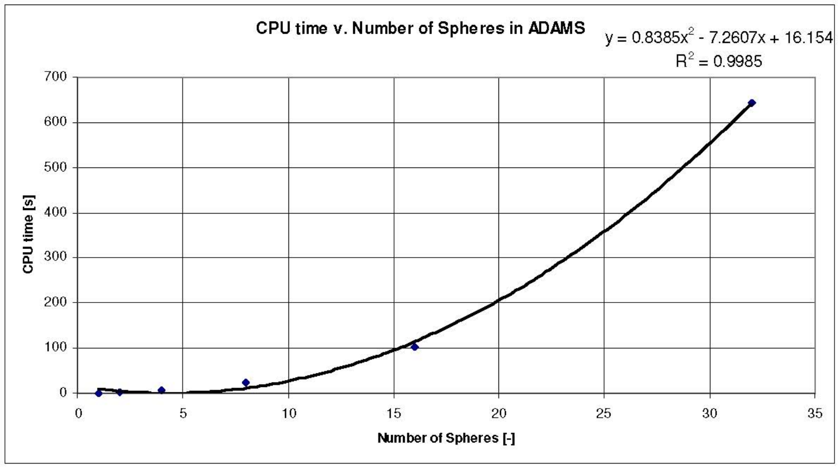
Table 1: Number of rigid bodies v. CPU time in ADAMS

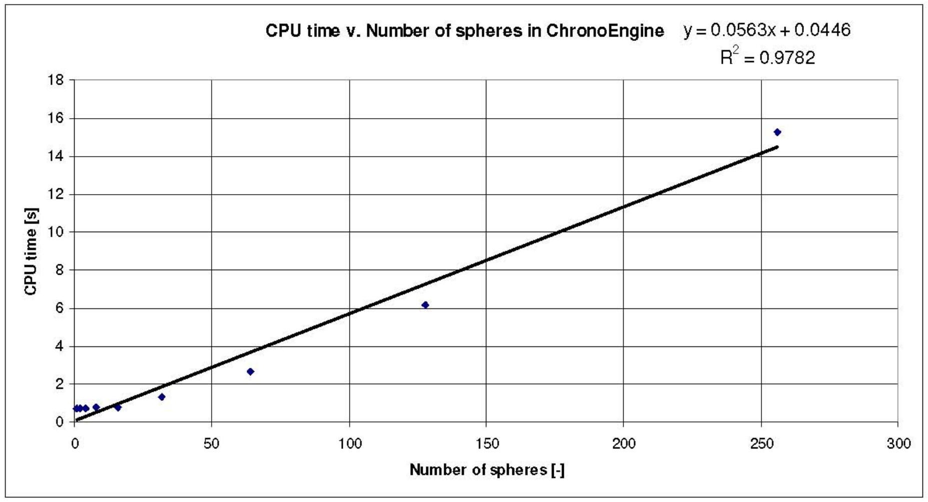
Number of Spheres	Max Number of Mutual Contacts [-]	CPU time (seconds)
1	1	0.41
2	3	3.3
4	14	7.75
8	44	25.36
16	152	102.78
32	560	644.4

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

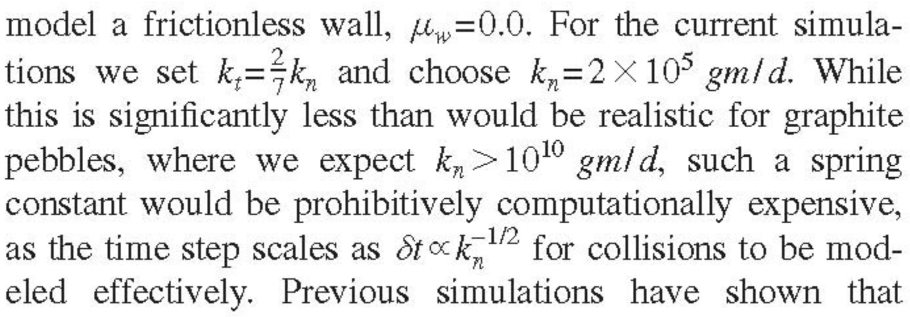


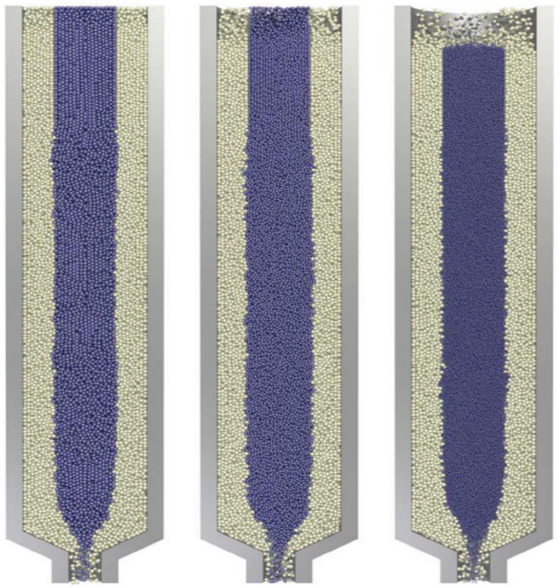


Time stepping has far better potential at efficiency

Smoothing (2) Simulating the PBR nuclear reactor

- Generation IV nuclear reactor with continuously moving fuel.
- Previous attempts: DEM methods on supercomputers at Sandia Labs regularization)
- 40 seconds of simulation for 440,000 pebbles needs 1 week on 64 processors dedicated cluster (Rycroft et al.)

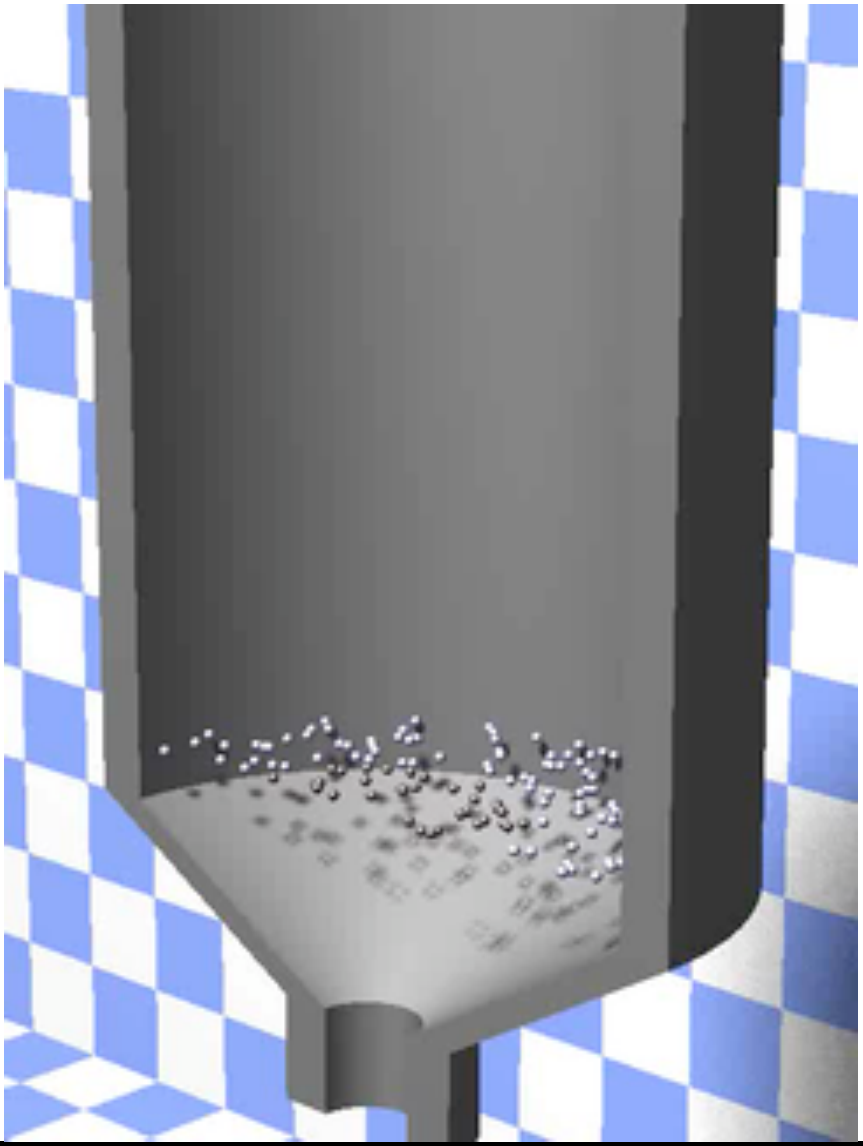

model a frictionless wall, $\mu_w=0.0$. For the current simulations we set $k_t=\frac{2}{7}k_n$ and choose $k_n=2\times10^5\ gm/d$. While this is significantly less than would be realistic for graphite pebbles, where we expect $k_n>10^{10}\ gm/d$, such a spring constant would be prohibitively computationally 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



Simulations with DEM. Bazant et al. (MIT and Sandia laboratories).

Simulating the PBR nuclear reactor

- 160’000 Uranium-Graphite spheres, 600’000 contacts on average
- ~Two millions dual variables.
- *1 CPU day on a single processor…*
- We estimate 3CPU days, compare with 150 CPU days for DEM !!!



Time Stepping – How did we get here?

- To make competitive with DEM, use relaxation and linearization-based stabilization, (convex QP with conic constraints):
- We show convergence, but did we destroy the predictive power?

$$
\begin{aligned}
M(\boldsymbol{v}^{(l+1)} - \boldsymbol{v}^{l}) &= \sum_{i\in\mathcal{A}(q^{(l)},\epsilon)} \left(\gamma_n^i \boldsymbol{D}_n^i + \gamma_u^i \boldsymbol{D}_u^i + \gamma_v^i \boldsymbol{D}_v^i\right) + \\
&\quad + \sum_{i\in\mathcal{G}_{\mathcal{B}}} \left(\gamma_b^i \nabla\Psi^i\right) + h\boldsymbol{f}_t(t^{(l)}, \boldsymbol{q}^{(l)}, \boldsymbol{v}^{(l)}) \\
0 &= \frac{1}{h}\Psi^i(\boldsymbol{q}^{(l)}) + \nabla\Psi^{i^T}\boldsymbol{v}^{(l+1)} + \frac{\partial\Psi^i}{\partial t}, \quad i \in \mathcal{G}_{\mathcal{B}} \\
0 &\le \frac{1}{h}\Phi^i(\boldsymbol{q}^{(l)}) + \nabla\Phi^{i^T}\boldsymbol{v}^{(l+1)} \boxed{-\mu^i\sqrt{(\boldsymbol{D}_u^{i,T}\boldsymbol{v})^2 + (\boldsymbol{D}_v^{i,T}\boldsymbol{v})^2}} \\
&\perp \quad \gamma_n^i \ge 0, \ i \in \mathcal{A}(q^{(l)},\epsilon) \\
(\gamma_u^i, \gamma_v^i) &= \operatorname{argmin}_{\mu^i\gamma_n^i \ge \sqrt{(\gamma_u^i)^2 + (\gamma_v^i)^2}} \quad i \in \mathcal{A}(q^{(l)},\epsilon) \\
&\quad \left[\boldsymbol{v}^T(\gamma_u \boldsymbol{D}_u^i + \gamma_v \boldsymbol{D}_v^i)\right] \\
\boldsymbol{q}^{(l+1)} &= \boldsymbol{q}^{(l)} + h\boldsymbol{v}^{(l+1)},
\end{aligned}
$$

(For small μ 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"]

Pause: what does convergence mean here?

We must now assign a meaning to

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

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

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

Pause(2) : What does convergence mean here?

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.

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

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

Further insight.

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

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

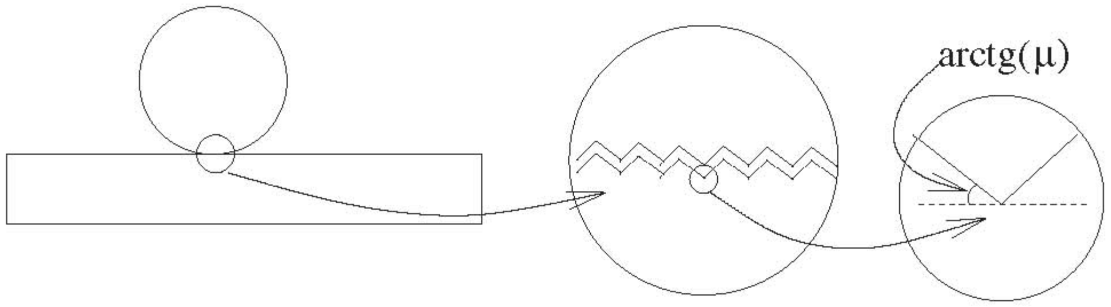
- If the time step is smaller than the variation in velocity then the gap function settles at

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

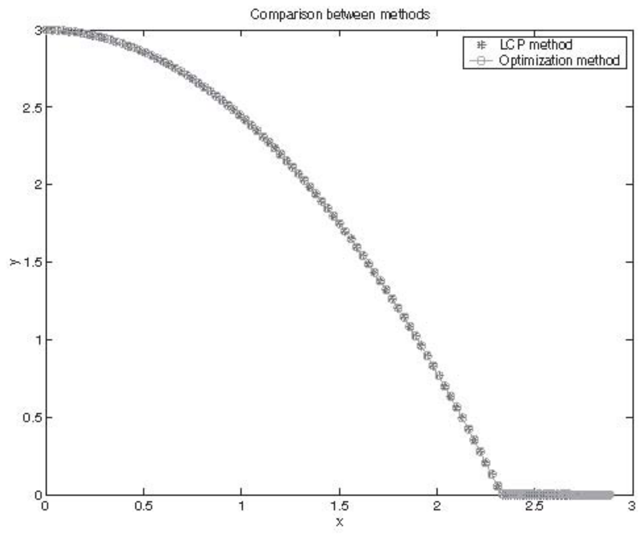
- So the solution is the same as the original scheme for a slightly perturbed gap function…..

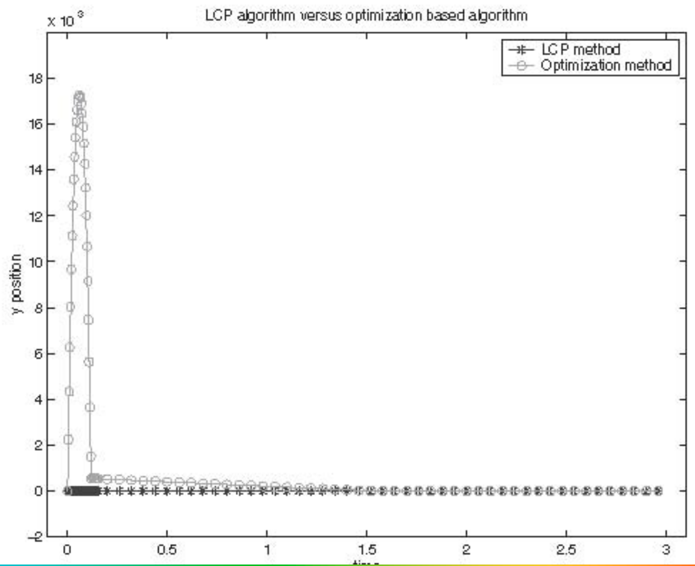
What is physical meaning of the relaxation?

- Origin



- Behavior





Cone complementarity

- Aiming at a more compact formulation:

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

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

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

Cone complementarity problem at each step

- 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:

$$\begin{aligned}
M(\boldsymbol{v}^{(l+1)} - \boldsymbol{v}^l) = & \sum_{i \in \mathcal{A}(q^{(l)},\epsilon)} (\gamma_n^i \boldsymbol{D}_n^i + \gamma_u^i \boldsymbol{D}_u^i + \gamma_v^i \boldsymbol{D}_v^i) + \\
& + \sum_{i \in \mathcal{G}_B} (\gamma_b^i \nabla \Psi^i) + h\boldsymbol{f}_t(t^{(l)}, \boldsymbol{q}^{(l)}, \boldsymbol{v}^{(l)}) \\
0 = & \frac{1}{h}\Psi^i(\boldsymbol{q}^{(l)}) + \nabla\Psi^{i^T} \boldsymbol{v}^{(l+1)} + \frac{\partial \Psi^i}{\partial t}, \quad i \in \mathcal{G}_B \\
0 \leq & \frac{1}{h}\Phi^i(\boldsymbol{q}^{(l)}) + \nabla\Phi^{i^T} \boldsymbol{v}^{(l+1)} \\
& \perp \quad \gamma_n^i \geq 0, \; i \in \mathcal{A}(q^{(l)}, \epsilon) \\
(\gamma_u^i, \gamma_v^i) = & \operatorname{argmin}_{\mu^i \gamma_n^i \geq \sqrt{(\gamma_u^i)^2 + (\gamma_v^i)^2}} \quad i \in \mathcal{A}(q^{(l)}, \epsilon) \\
& [\boldsymbol{v}^T (\gamma_u \boldsymbol{D}_u^i + \gamma_v \boldsymbol{D}_v^i)]
\end{aligned}$$

becomes..

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

This is a CCP,
CONE COMPLEMENTARITY PROBLEM

Cone complementarity—Decomposable cones.

- Here we introduced the convex cone

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

$\mathcal{FC}^i$	In R^3 is *i*-th friction cone
$\mathcal{BC}^i$	is R

- ..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: $(N\gamma_\mathcal{E} + \boldsymbol{r}) \in -\Upsilon^\circ \quad \perp \quad \gamma_\mathcal{E} \in \Upsilon$

General: The iterative method (Anitescu-Tasora 09).

- How to efficiently solve the Cone Complementarity Problem for large-scale systems?

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

- Our method: use a projected iteration (Gauss-Seidel)

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

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

$$B^{r} = \begin{bmatrix} \eta_1 I_{n_1} & 0 & \cdots & 0 \\ 0 & \eta_2 I_{n_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \eta_{n_k} I_{n_{n_k}} \end{bmatrix}$$

$$N^{T} = \begin{bmatrix} 0 & K_{12} & K_{13} & \cdots & K_{1n_k} \\ 0 & 0 & K_{23} & \cdots & K_{2n_k} \\ 0 & 0 & 0 & \cdots & K_{3n_k} \\ \vdots & \vdots & \cdots & \ddots & \vdots \\ 0 & 0 & 0 & & 0 \end{bmatrix}$$

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

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 $(x^{r+1} - x^r)^T \left((\lambda \omega B^r)^{-1} + K^r - \frac{N}{2} \right) (x^{r+1} - x^r) \geq \beta \left\| x^{r+1} - x^r \right\|^2$.

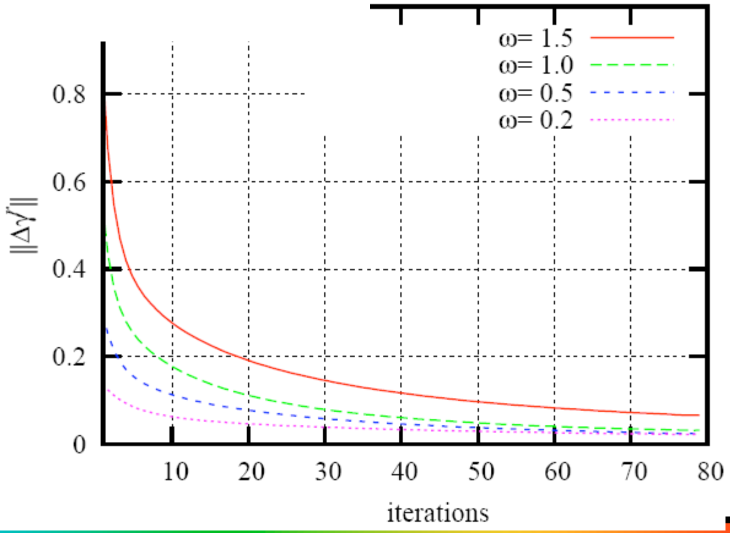
Always satisfied in multibody systems

Essentially free choice, we use identity blocks

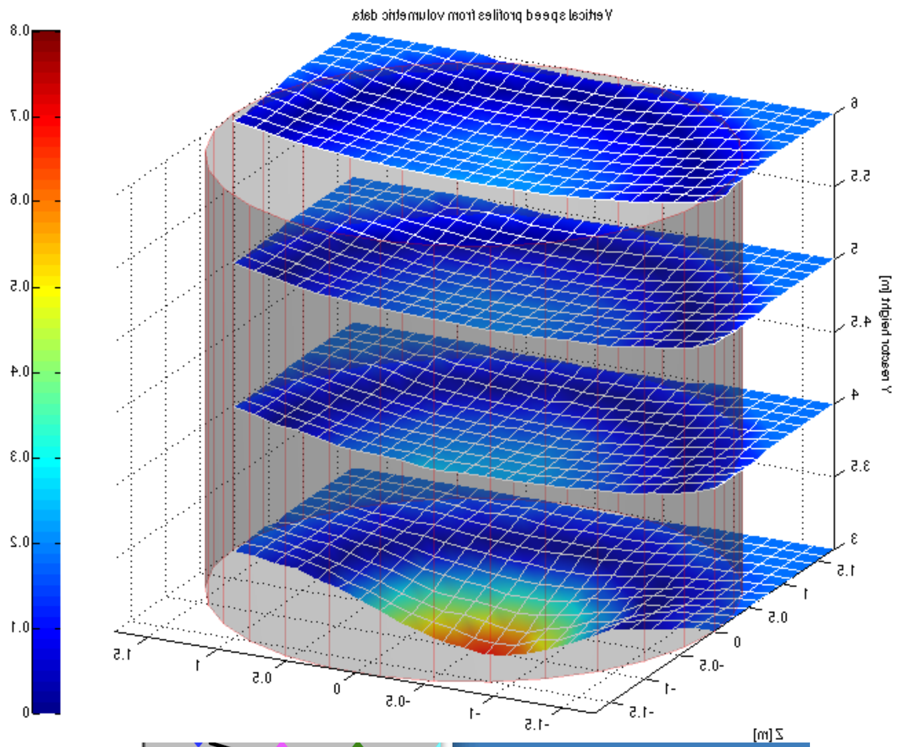
Use ω overrelaxation factor to adjust this

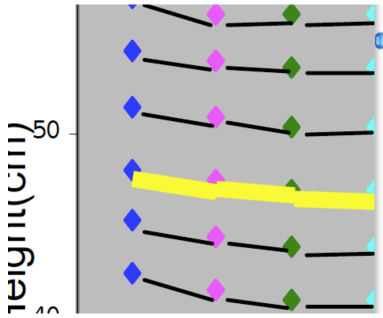
- Under the above assumptions, we can prove THEOREMS about convergence.

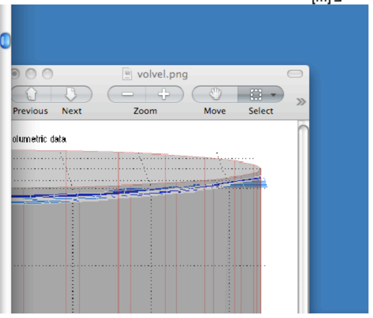
- The method produces a bounded sequence with an unique accumulation point.

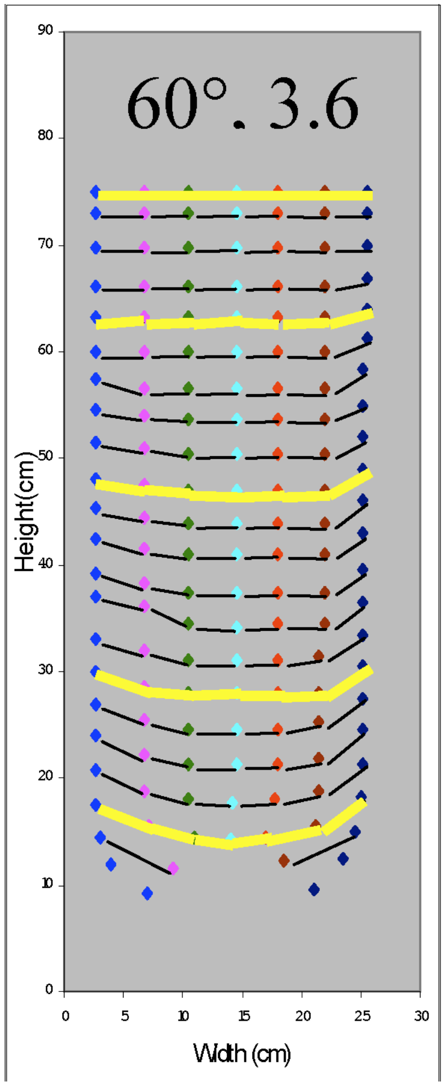


Validation of projected GS convex relaxation time-stepping: PBR

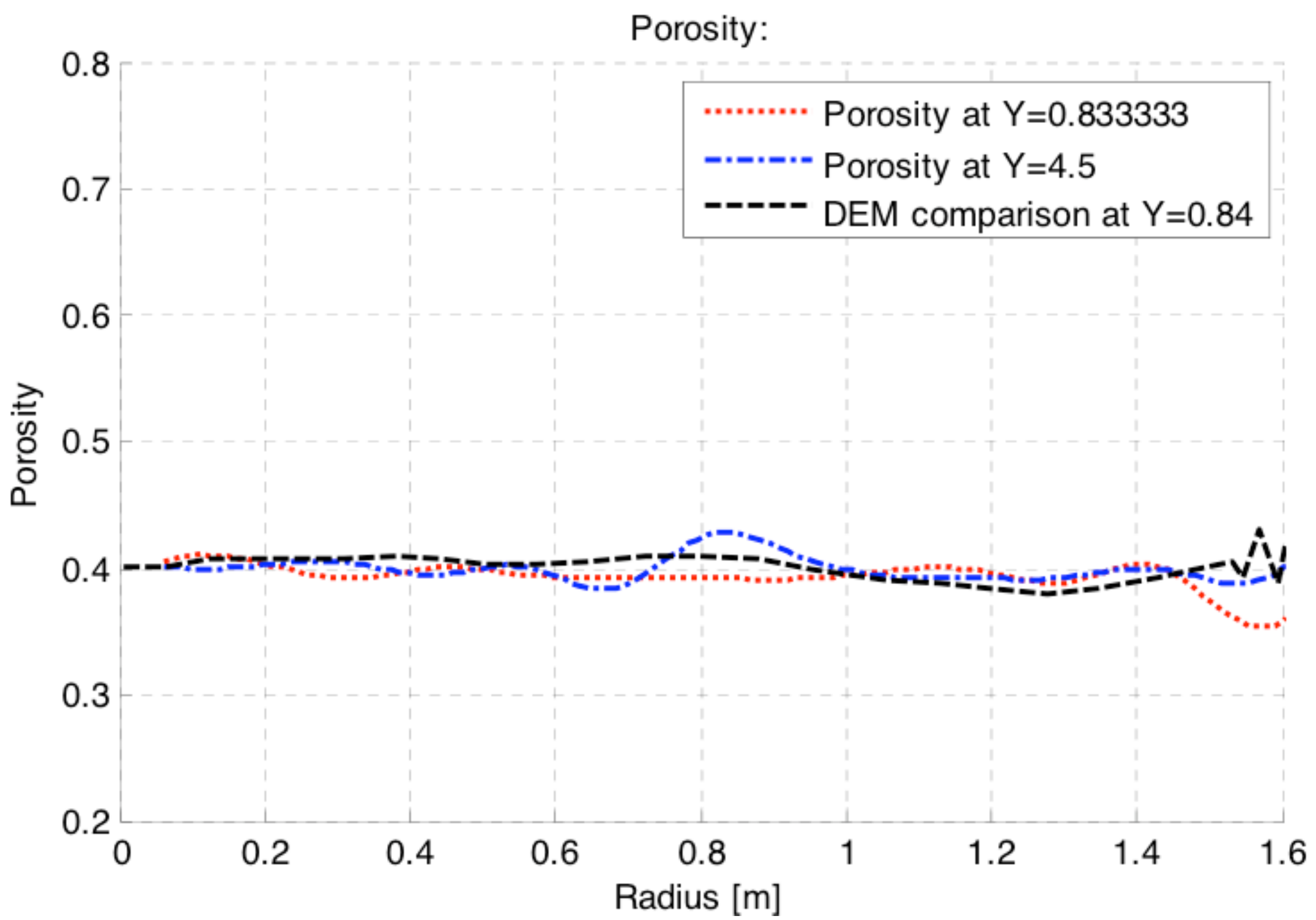




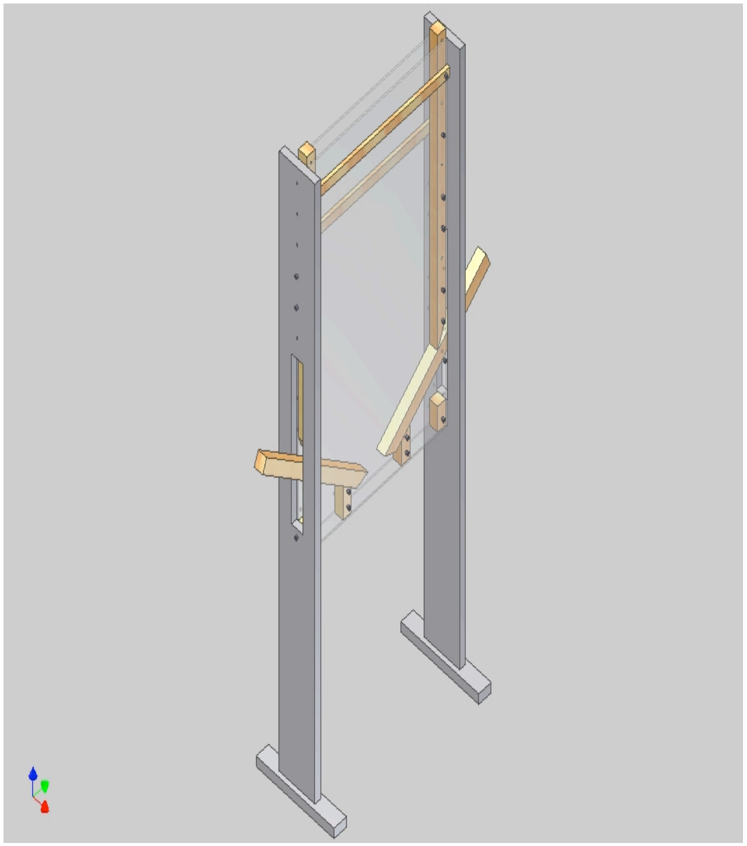


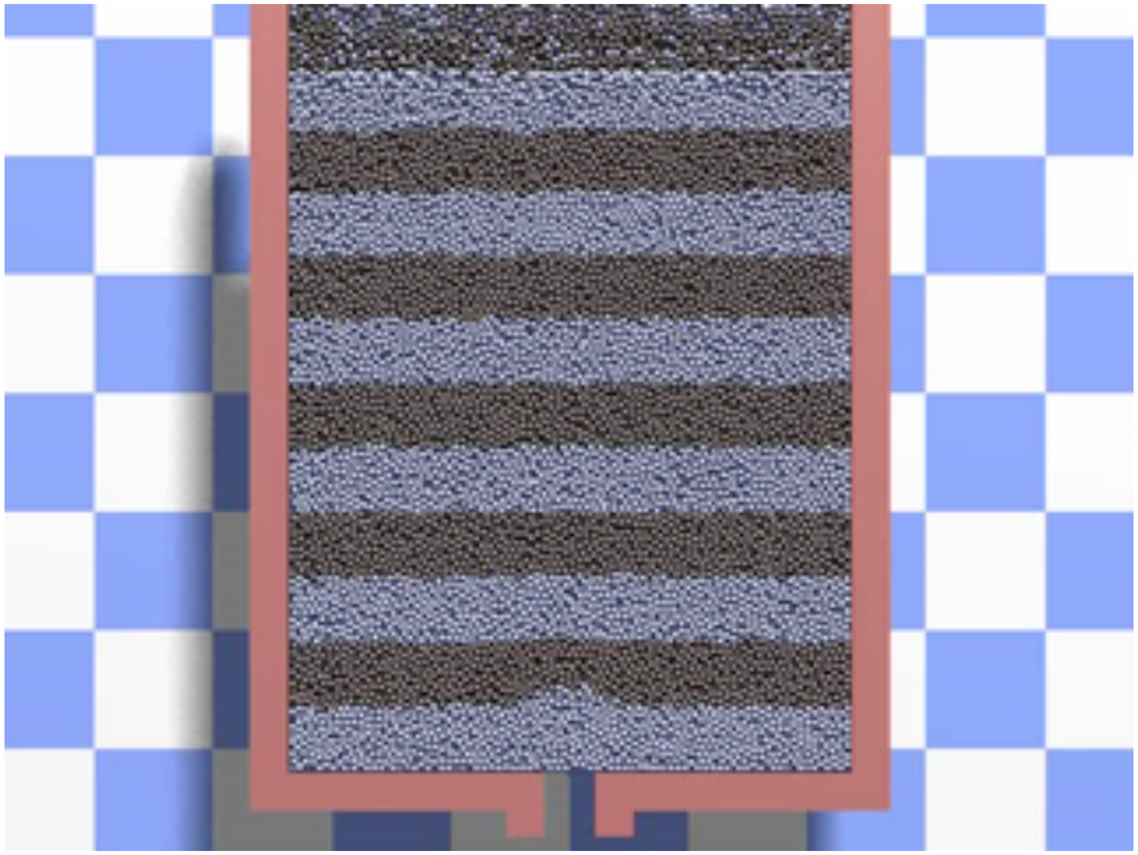


Validation: PBR: Packing statistics (T & A, 09)

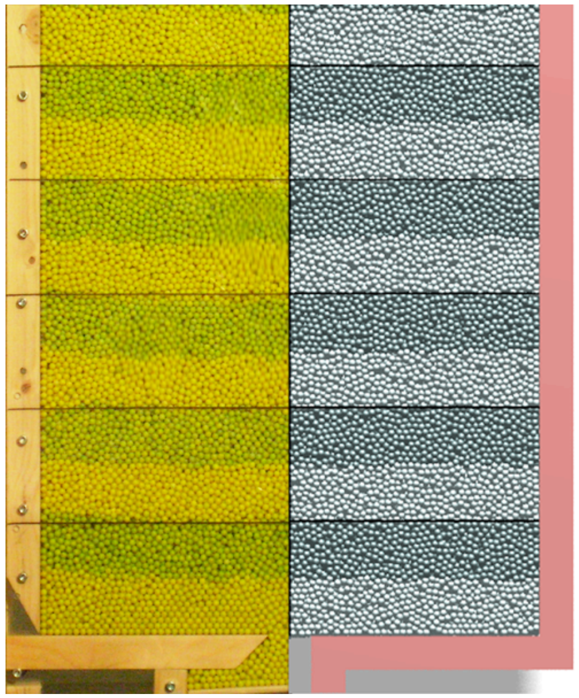


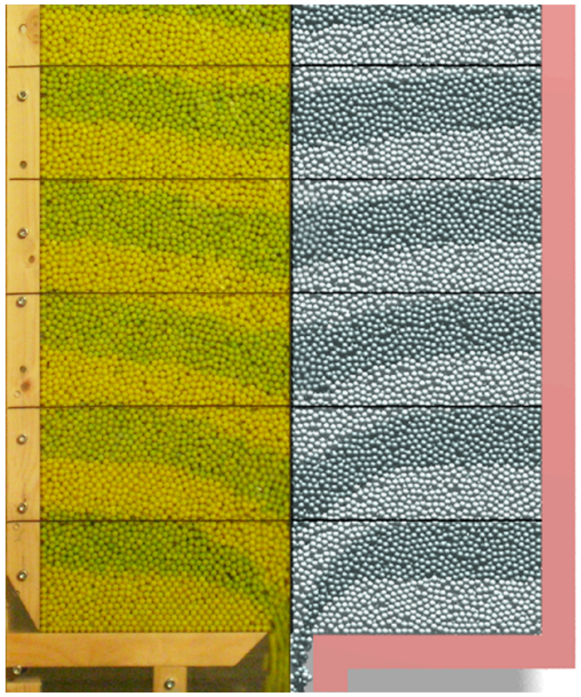
DVI: Time-stepping validation : Hopper (Tasora, Anitescu, 2009)

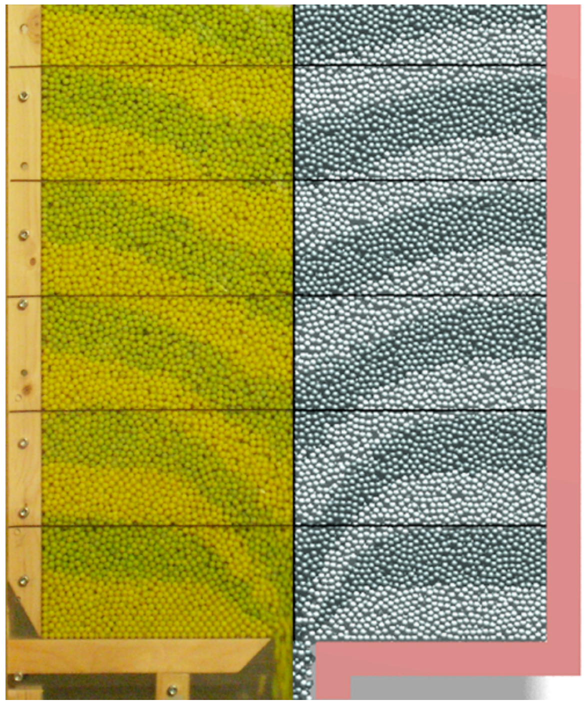




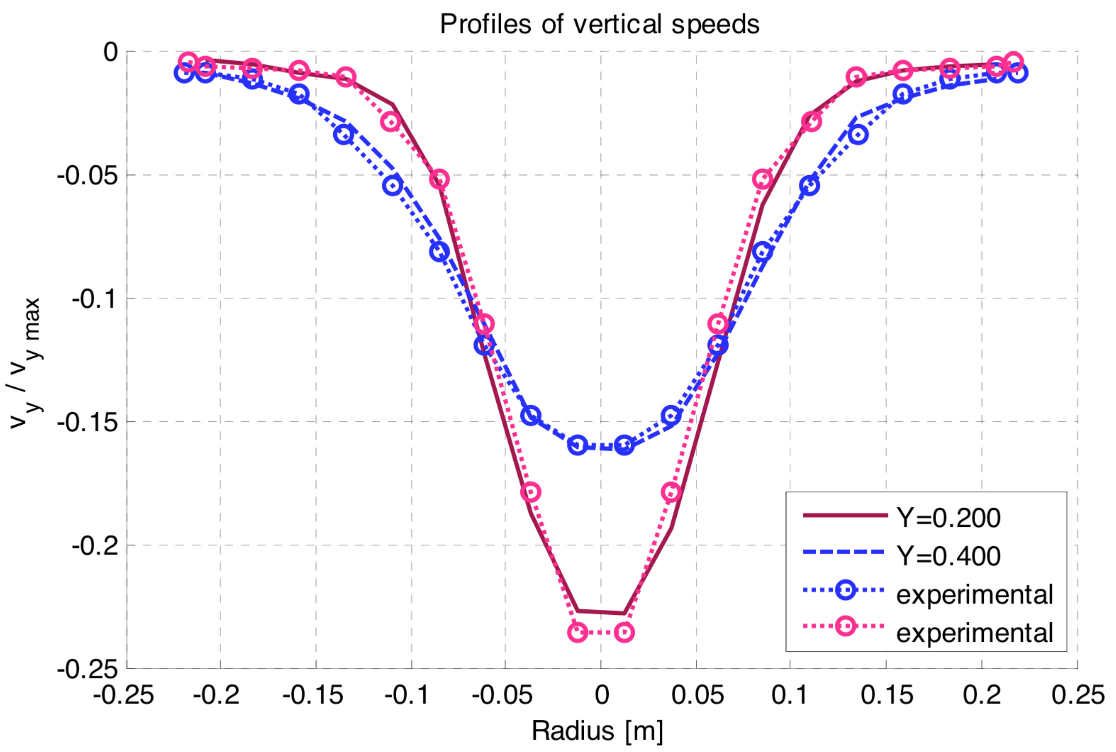
Hopper experiment and simulation: Images


t=0s


t=0.6s


t=1.2s

Hopper Results: Velocity



- Note that there are sphere measurement errors of 2%, and particle-wall friction variations of 10% (reduced by climate control).

Future work

- N non symmetric, but positive semidefinite.
- Parallelizing the algorithms: block Jacobi with Gauss Seidel blocks, or coloring GS (50% there).
- Huge scale simulation. Multigrid for rigid multibody dynamics?
- Including a good collision model– here we are at a loss with rigid body theory – may need some measure of deformability – convolution complementarity.
- Involving other physics … fluid flow.
- Compare with experimental data.

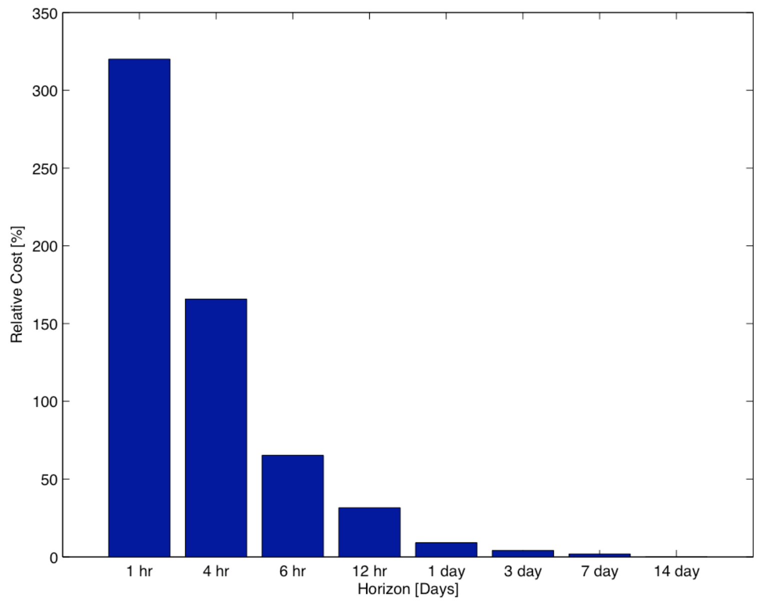
Conclusions

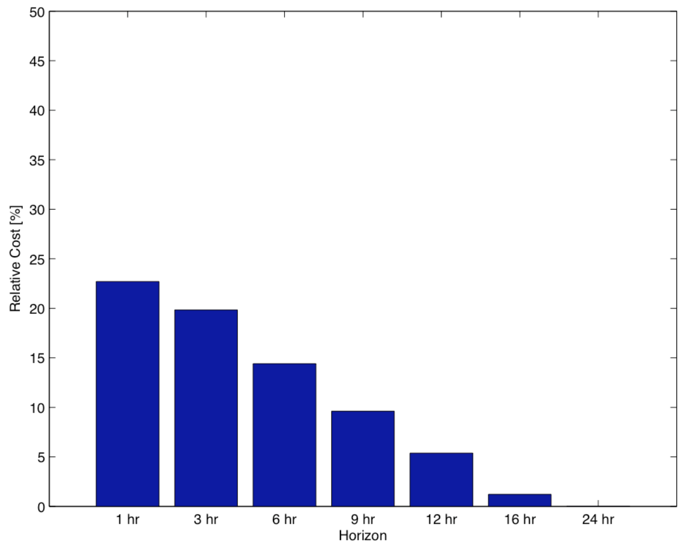
- We have defined a new algorithm for complementarity problems with conic constraints, that solves CONVEX subproblems.
- We have shown that it can solve very large problems in granular flow far faster than DEM.
- It is the first iterative algorithm that provably converges for nonsmooth rigid body dynamics.
- We have shown that it correctly computes the statistics of dense granular flow by validation with PHYSICAL experiment

What are roadblocks to DVI progress?

- Need to extend the use of time-stepping methods – new applications and new extensions.
- Need to accommodate heterogeneous systems, that may have sharply different time scales – multirate methods
- Need to efficiently solve very large scale problems 10^7-10^9 variables.
  - Use semismooth Newton methods to exploit the high time coherence of the solution.
  - Use domain decomposition and multigrid methods for efficiently solving the linear systems.

New applications: Stochastic Optimization of Integrated Energy Systems under Uncertainty





- Stochastic Optimization of hybrid systems with weather uncertainty info can result in enormous cost and energy savings (Zavala, Anitescu, et al, 2009).
- But a SO problem must be solved in real-time in nonlinear model predictive control (NLMPC).

DVI for real-time model predictive control

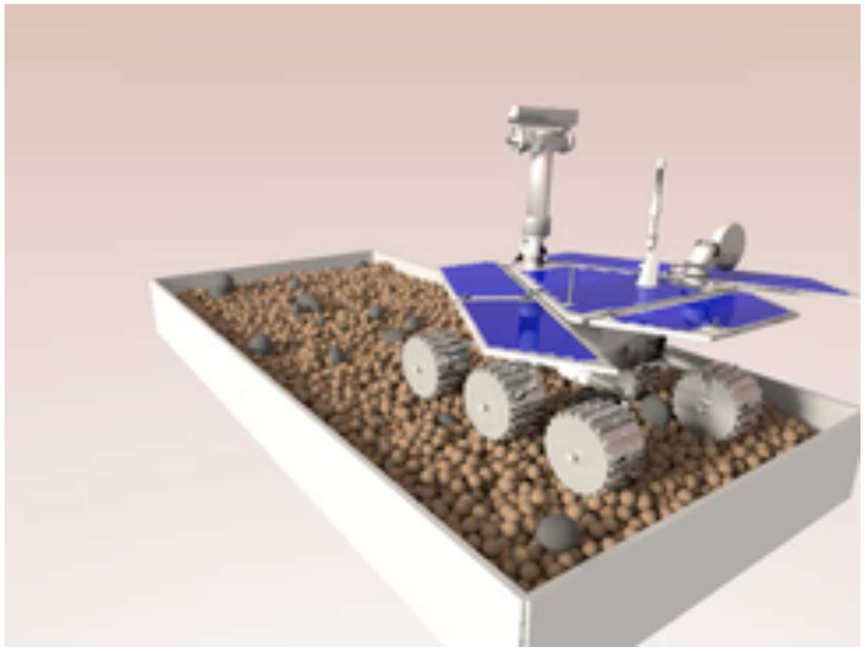
- The problem to be solved is:

$$
\begin{aligned}
\min_{u(\tau)} \ \mathbf{E}_{\chi(\tau)\in\Omega_k} \left[\int_{t_k}^{t_k+T} \varphi(z(\tau), y(\tau), u(\tau), \chi(\tau))d\tau\right] \\
\frac{dz}{d\tau} = \mathbf{f}\left(z(\tau), y(\tau), u(\tau), \chi(\tau)\right) \\
0 = \mathbf{g}(z(\tau), y(\tau), u(\tau), \chi(\tau)) \\
0 \geq \mathbf{h}(z(\tau), y(\tau), u(\tau), \chi(\tau)) \\
z(t_k) = x_k, \quad \tau \in [t_k, t_k + T], \quad \chi(\tau) \in \Omega_k,
\end{aligned}
$$

- But the optimal control u can be obtained by sensitivity analysis and results in a DVI!
- Current real-time approaches do smoothing, we expect time-stepping will be faster, as in the granular case.

Solving DVIs with sharply different time scales

- E.g. special physics pebble in PBR or rover on Mars.
- It is inefficient to run the simulation at the time scale of the rover, the advantage of time-stepping methods is lost.
- We propose to investigate a multirate algorithm for DVI.



A predictor-corrector multirate approach.

- Assume a separation of variables, in regular and important ('fast') variables: $y = (y^r, y^f), x = (x^r, x^f)$
- Advance all variables by a "large" time step H, and interpolate the regular variables between the interval endpoints, creating $\tilde{x}^r(t), \tilde{y}^r(t), t \in [t^i, t^i + H]$
- In the corrector step, solve the reduce problem for the important variables, with much smaller time step, h.

$$\begin{aligned} \frac{dy^f}{dt} &= f^f(t, (y^f(t), \tilde{y}^r(t)), (x^f(t), \tilde{x}^r(t))) \\ x^f(t) &\in SOL(K^f; F^f(t, (y^f(t), \tilde{y}^r(t)), (\cdot, \tilde{x}^r(t)))) \\ y^f(t^i) &= y^{f,H,i}. \end{aligned}$$

Applied mathematics questions -- Multirate.

- How do we choose the fast and slow variables.
- Is the predictor-corrector scheme convergent? We plan to investigate this using perturbation theory of VI.
- What is a good interpolation scheme that results in minimum violation of the constraints in the corrector step?
- We point out that convergence results for multirate methods for ODE and PDE cannot be extended due to the existence of the inequality constraints – even the DAE case is not completely solved

The time-stepping subproblem

- Its most common form: cone complementarity problem

$$s_i = (Nx+q)_i \in K^{i,*} \subset \mathbb{R}^{n_i}, x_i \in K^i \subset \mathbb{R}^{n_i},\ \langle x_i, s_i \rangle = 0,\ i = 1, 2, \ldots, m_K.$$

- It is a generalization of optimization over cones.
- The cones of interest are $\mathbb{R}^+, \mathbb{R}$ or low-dimensional second-order cone.

$$\mathcal{K}_n = \left\{ (a_1, a_2, \ldots, a_n) \in \mathbb{R}^+ \times \mathbb{R}^{n-1} | a_1 \geq \sqrt{\sum_{i=2}^{n} a_i^2} \right\}, \quad n \geq 2.$$

How to solve very large DVI instances by time-stepping?

- The solution of the time-stepping subproblem has a high time coherence.
- Therefore it will be useful to easily restart it from the solution of the past problem “warm-start” it.
- Interior point with CHOLMOD (Petra et al., OMS, 09) has shown important potential for this problem.
- Nonetheless, interior-point approaches are hard to warm-start. In addition, the difficulty in solving the Newton steps with iterative methods due to scaling issues, have pointed us towards semismooth methods.

Semismooth methods for large-scale DVI

- Rephrase as semismooth nonlinear equation-Newton.

$$0 = \Phi(x) = \left(\begin{array}{c} \psi_1(x_1, (Nx+q)_1) \\ \psi_2(x_2, (Nx+q)_2) \\ \vdots \\ \psi_{m_K}(x_{m_K}, (Nx+q)_{m_K}) = 0 \end{array} \right) \quad \Rightarrow \quad \begin{array}{c} Vd + \Phi(y) = 0 \\ \forall V \in \partial_B \Phi(y) \end{array}$$

$$\left. \begin{array}{r} \psi_i : \mathbb{R}^{n_i} \times \mathbb{R}^{n_i} \to \mathbb{R}^{n_i}, \\ y^1, y^2 \in \mathbb{R}^{n_i}, \ \psi_i(y^1, y^2) = 0, \end{array} \right\} \Leftrightarrow y^1 \in K^i, y^2 \in K^{i,*} \ \langle y^1, y^2 \rangle = 0.$$

- Use Jordan-algebra-based Fischer-Burmeister functions, in the case of second-order cones (B-subdifferential from Pan 08) .

$$(a_1, a_2, a_3) \circ (b_1, b_2, b_3) = (a_1 b_1 + a_2 b_2 + a_3 b_3, a_1 b_2 + b_1 a_2, a_1 b_3 + b_1 a_3),$$

$$\psi_i(y^1, y^2) = y^1 + y^2 - \left((y^1 \circ y^1) + ((y^2 \circ y^2) \right)^{\circ \frac{1}{2}} .$$

Applied Math Questions -- Semismooth

- The key difference is the existence of the second-order cone
- If N is symmetric, can we obtain a symmetric system that does not have conditioning problems (Extend the Benson – Munson method to cone case).
- Can it still be warm-started well?
- What is an appropriate globalization strategy ?
- This method will be implemented in TAO.

Solving the large-scale linear system

- We use methods inspired by PDEs, since in many DVIs the system has macroscale coherence, but the macroscale constitutive laws are not know.
- For dense granular flow DVI, many times the problem does homogenize after about 5 atomic layers to an unknown parabolic-like behavior. We use domain decomposition.
- For DVIs with different co-existing regimes, and different macroscale variables, we use adaptive multigrid approaches, based on compatible relaxation.

Solving the large-scale linear equations: domain decomposition

- Robust first step for extending the size of the problems resolved efficiently.
- We will investigate both additive Schwarz preconditioning approaches, as well as nonlinear domain decomposition, with subdomains $D_1, D_2, \ldots, D_d$. and metadomains $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_m$.

$$C = \sum_{j=1}^{d} (P_{D_1} V P_{D_1}^T)^{-1}, \qquad \min_{x_i \in K^i, i \in D_j, D_j \in \mathcal{D}_k} \frac{1}{2} x^T N x + q^T x$$

- AM issues: size of domain overlap, partitioning strategies, subproblem resolution precision.

Large-scale linear problem– multigrid.

- For adaptative and different macroscale variable problems: compatible-relaxation (CR)
- CR: GS while keeping the coarse variable constrained.

$$Vd + \Phi(y) = 0, \quad x_k^c = \sum \mu_{ki} x_i, \quad k \in \mathcal{I}^c$$

- Use speed of convergence of CR do decide which variables should be kept in the coarse level.
- Solve an algebraic subproblem to determine the coarsening rules and interpolation operator.

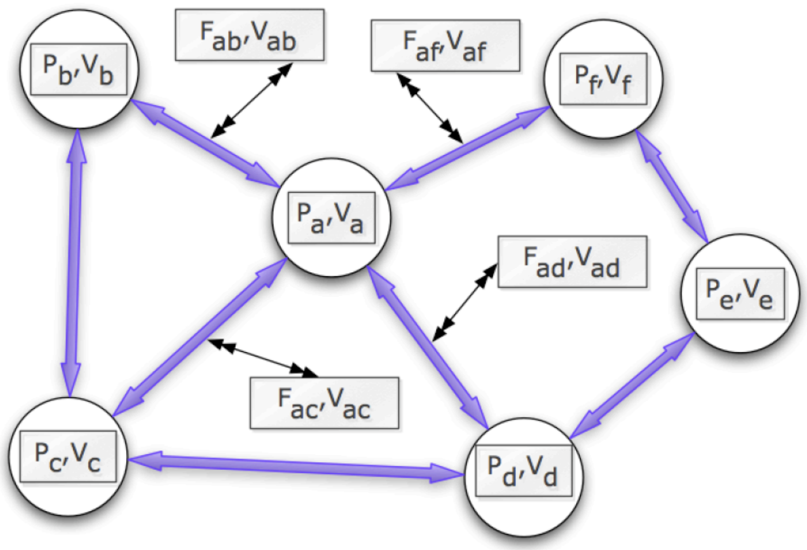
$$\begin{array}{lll} \min_{x_i, i \in \Omega_k} & & \Psi(x) \\ & \text{subject to} & x_l^c = \sum \mu_{li} x_i, \quad l \in \Omega_k^{c,-}. \quad (\Omega_k^c) \\ & & \sum_i V_{ij} x_j = q_i, \; i \in \Omega_k^I. \end{array}$$

- The impressive results for discrete state problems of CR AMG recommends it despite its risks.

A graph-based representation for DVI.

- The usual abstraction where function and Jacobian information is computed by the user seems difficult to manipulate optimally, particularly in environments where there is a premium on communication.
- It would be helpful if the representation naturally allows modeling of the problem and problem resolution in the same memory space.
- We propose to restrict our attention to problems that have a graph-like incidence structure, which covers all our motivating cases.
- Since the type of functions and constraints attached to an edge is very small, minimal information needs to be passed around, except for the incidence info.

A graph-based representation for DVI:



Differential equation for node a:

$$\begin{aligned} m^a \frac{d\dot{q}^a}{dt} &= k^a + \textstyle\sum_{\theta=b,c,d,f} \gamma^{a\theta} n(q^a, q^\theta) \\ &\quad + \textstyle\sum_{\theta=b,c,d,f} \beta^{a\theta} t(q^a, q^\theta) \end{aligned}$$

Constraints for edges involving a.

$$\begin{aligned} w^{a\theta} &= t(q^a, q^\theta)(\dot{q}^a - \dot{q}^b)) \\ \begin{pmatrix} \gamma^{a\theta} \\ \beta^{a\theta} \end{pmatrix} \in K \quad &\perp \begin{pmatrix} \mu^{a\theta} \left| w^{a\theta} \right| \\ w^{a\theta} \end{pmatrix} \in K^*, \\ \theta &= b, c, d, f \end{aligned}$$

Figure 2.4: Graph representation and equations generated for a two-dimensional dense granular flow configuration. Here $P_u = \{m^u, k^u\}$, and, respectively $F_{vw} = \{\mu^{vw}\}$, are parameters and $V_u = \{q^u, \dot{q}^u\}$, and, respectively, $V_{vw} = \{\gamma^{vw}, \beta^{v,w}\}$ are variables attached to a node u and, respectively, an edge vw. The equations are generated using the formulation in [5], [20].

Milestones

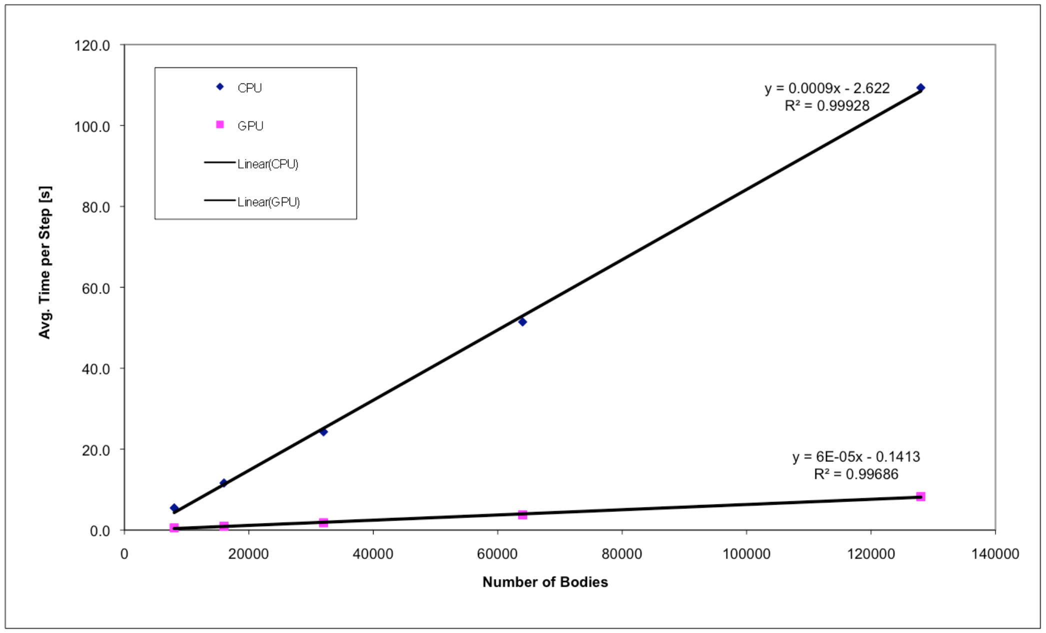
- Year 1. Development of the Graph-based DVI representation framework (§2.6). Semismooth Newton algorithm design, analysis, and prototyping (§2.4.1). Givens 1: Graph-based DVI benchmark examples development (§2.6). Givens 2: DVI modeling of NLMPC (§2.6).

- Year 2. Semismooth algorithm implementation in TAO, using template from [27] for the complementarity case (§2.4.1). Design and implementation of domain decomposition approaches using TAO/PETSC (§2.4.2). Multirate algorithm convergence analysis and prototyping (§2.5). Givens 1: Granular dynamics modeling extensions development and testing (§2.6). Givens 2: Testing, benchmarking and reporting of semismooth and domain decomposition methods (§2.4.2 and §2.4.1).

- Year 3. Design and prototyping of the multigrid method (§2.4.3). Development, implementation and testing of multigrid and multirate methods (§2.4.3 and §2.5). Givens 1: Multigrid testing and benchmarking (§2.4.3). Givens 2: Multirate examples development and testing. (§2.5).

DVI accomplishment highligh: GPU implementation

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



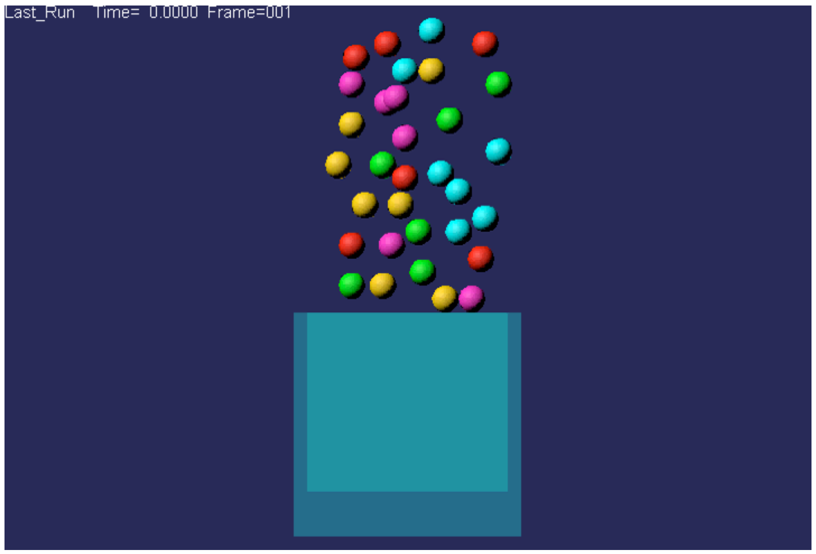
	CPU			GPU					
Number of Bodies	CCP Time	CD Time	Step Time	CCP Time	CD Time	Step Time	CCP Speedup	CD Speedup	Step Speedup
128000	103.6665 8	3.80176	109.3428 7	6.97682	0.74488	8.27050	14.8587	5.1038	13.2208

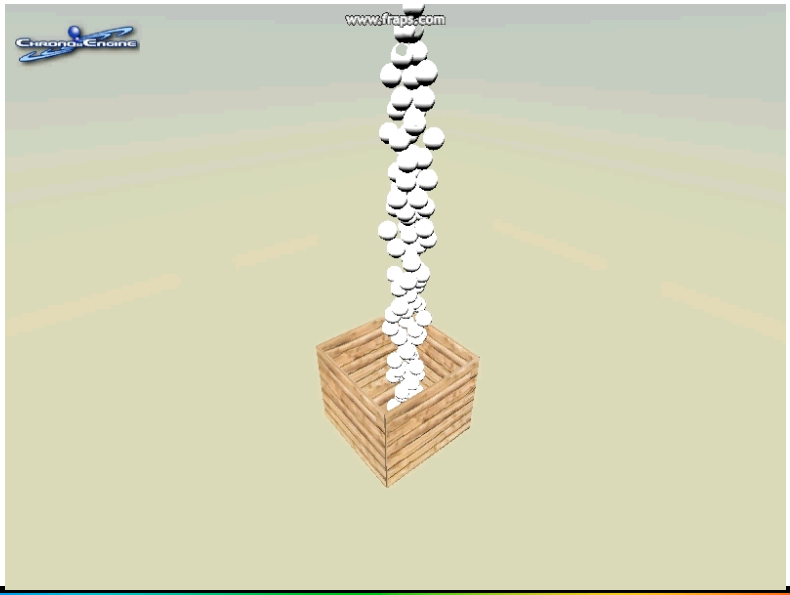
Proposed Research Outline: Advanced Methods for DVI.

- DVI: Motivating applications and comparison with regularization methods-DEM.
- DVI: validation of time-stepping schemes.
- Research needs and proposed solutions.
- Implementation framework.

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

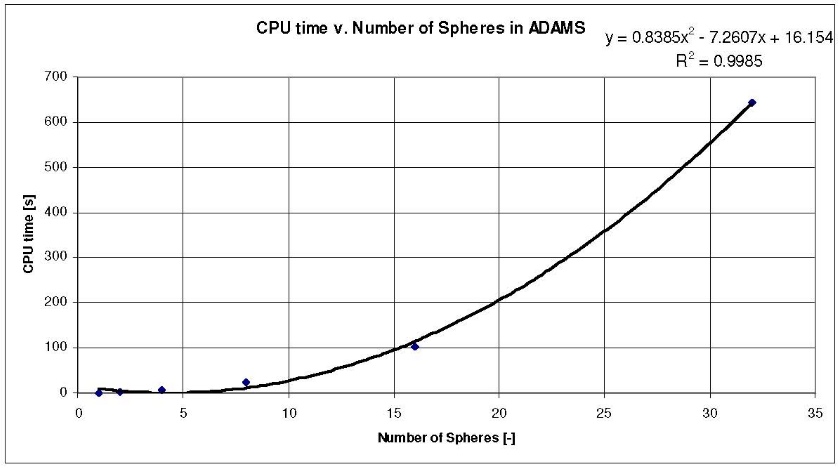
Table 1: Number of rigid bodies v. CPU time in ADAMS

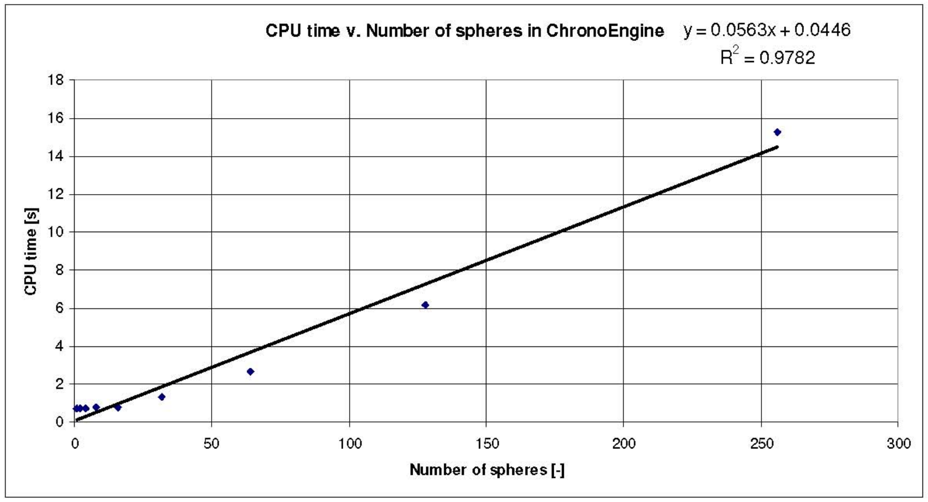
Number of Spheres	Max Number of Mutual Contacts [-]	CPU time (seconds)
1	1	0.41
2	3	3.3
4	14	7.75
8	44	25.36
16	152	102.78
32	560	644.4

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





Conclusion 1: Often, time stepping is more promising,

Time stepping scheme -- original

- A measure differential inclusion solution can be obtained by time-stepping (Stewart, 1998, Anitescu 2006)

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

Speeds

Reaction impulses

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

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

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

Contact constraint equations

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

COMPLEMENTARITY!

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

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

Coulomb 3D friction model

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

Time Stepping -- Convex Relaxation

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

$$
\begin{aligned}
M(\boldsymbol{v}^{(l+1)} - \boldsymbol{v}^{l}) &= \sum_{i\in\mathcal{A}(q^{(l)},\epsilon)} \left(\gamma_n^i \boldsymbol{D}_n^i + \gamma_u^i \boldsymbol{D}_u^i + \gamma_v^i \boldsymbol{D}_v^i\right) + \\
&\quad + \sum_{i\in\mathcal{G}_{\mathcal{B}}} \left(\gamma_b^i \nabla\Psi^i\right) + h\boldsymbol{f}_t(t^{(l)}, \boldsymbol{q}^{(l)}, \boldsymbol{v}^{(l)}) \\
0 &= \frac{1}{h}\Psi^i(\boldsymbol{q}^{(l)}) + \nabla\Psi^{i^T}\boldsymbol{v}^{(l+1)} + \frac{\partial\Psi^i}{\partial t}, \quad i \in \mathcal{G}_{\mathcal{B}} \\
0 &\le \frac{1}{h}\Phi^i(\boldsymbol{q}^{(l)}) + \nabla\Phi^{i^T}\boldsymbol{v}^{(l+1)} \boxed{-\mu^i\sqrt{(\boldsymbol{D}_u^{i,T}\boldsymbol{v})^2 + (\boldsymbol{D}_v^{i,T}\boldsymbol{v})^2}} \\
&\perp \quad \gamma_n^i \ge 0, \; i \in \mathcal{A}(q^{(l)}, \epsilon) \\
(\gamma_u^i, \gamma_v^i) &= \operatorname{argmin}_{\mu^i\gamma_n^i \ge \sqrt{(\gamma_u^i)^2 + (\gamma_v^i)^2}} \quad i \in \mathcal{A}(q^{(l)}, \epsilon) \\
&\qquad \left[\boldsymbol{v}^T(\gamma_u \boldsymbol{D}_u^i + \gamma_v \boldsymbol{D}_v^i)\right] \\
\boldsymbol{q}^{(l+1)} &= \boldsymbol{q}^{(l)} + h\boldsymbol{v}^{(l+1)},
\end{aligned}
$$

(For small μ 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"]

Cone complementarity

- Aiming at a more compact formulation:

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

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

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

Cone complementarity problem at each step

- 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:

$$\begin{aligned}
M(\boldsymbol{v}^{(l+1)} - \boldsymbol{v}^l) = & \sum_{i \in \mathcal{A}(q^{(l)},\epsilon)} (\gamma_n^i \boldsymbol{D}_n^i + \gamma_u^i \boldsymbol{D}_u^i + \gamma_v^i \boldsymbol{D}_v^i) + \\
& + \sum_{i \in \mathcal{G}_B} (\gamma_b^i \nabla \Psi^i) + h\boldsymbol{f}_t(t^{(l)}, \boldsymbol{q}^{(l)}, \boldsymbol{v}^{(l)}) \\
0 = & \frac{1}{h}\Psi^i(\boldsymbol{q}^{(l)}) + \nabla \Psi^{i^T} \boldsymbol{v}^{(l+1)} + \frac{\partial \Psi^i}{\partial t}, \quad i \in \mathcal{G}_B \\
0 \leq & \frac{1}{h}\Phi^i(\boldsymbol{q}^{(l)}) + \nabla \Phi^{i^T} \boldsymbol{v}^{(l+1)} \\
& \perp \quad \gamma_n^i \geq 0, \ i \in \mathcal{A}(q^{(l)}, \epsilon) \\
(\gamma_u^i, \gamma_v^i) = & \operatorname{argmin}_{\mu^i \gamma_n^i \geq \sqrt{(\gamma_u^i)^2 + (\gamma_v^i)^2}} \quad i \in \mathcal{A}(q^{(l)}, \epsilon) \\
& [\boldsymbol{v}^T (\gamma_u \boldsymbol{D}_u^i + \gamma_v \boldsymbol{D}_v^i)]
\end{aligned}$$

becomes..

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

This is a CCP,

CONE COMPLEMENTARITY PROBLEM

Cone complementarity—Decomposable cones.

- Here we introduced the convex cone

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

$\mathcal{FC}^i$ In R^3 is *i*-th friction cone

$\mathcal{BC}^i$ is R

- ..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: $(N\gamma_{\mathcal{E}} + \boldsymbol{r}) \in -\Upsilon^\circ \quad \perp \quad \gamma_{\mathcal{E}} \in \Upsilon$

General: The iterative method

- Question 3: 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

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

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

$$B^{r} = \begin{bmatrix} \eta_1 I_{n_1} & 0 & \cdots & 0 \\ 0 & \eta_2 I_{n_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \eta_{n_k} I_{n_{n_k}} \end{bmatrix}$$

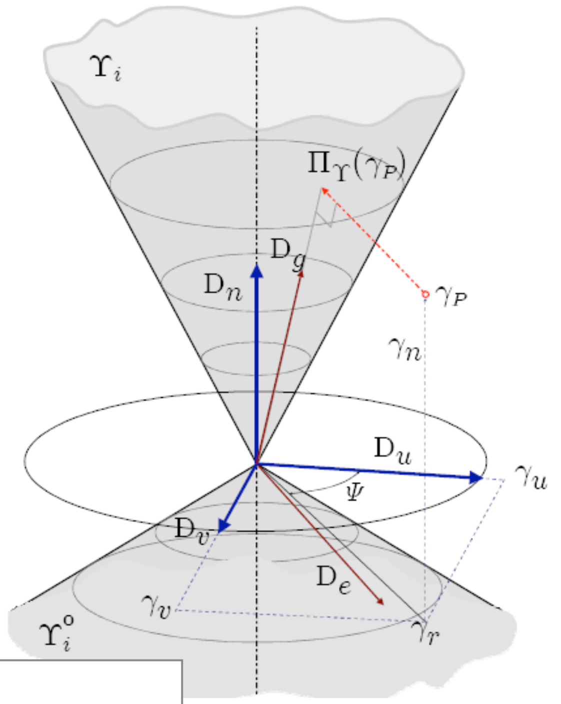
$$N^{T} = \begin{bmatrix} 0 & K_{12} & K_{13} & \cdots & K_{1n_k} \\ 0 & 0 & K_{23} & \cdots & K_{2n_k} \\ 0 & 0 & 0 & \cdots & K_{3n_k} \\ \vdots & \vdots & \cdots & \ddots & \vdots \\ 0 & 0 & 0 & & 0 \end{bmatrix}$$

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

The projection operator is easy and separable

- *For each frictional contact constraint:*

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



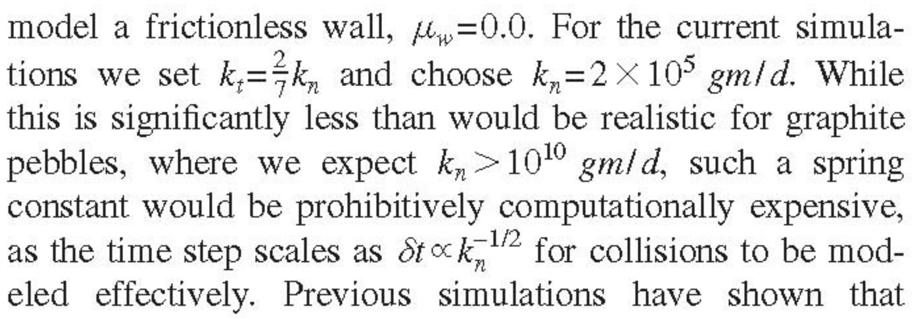
- *For each bilateral constraint, simply do nothing.*
- The complete operator:

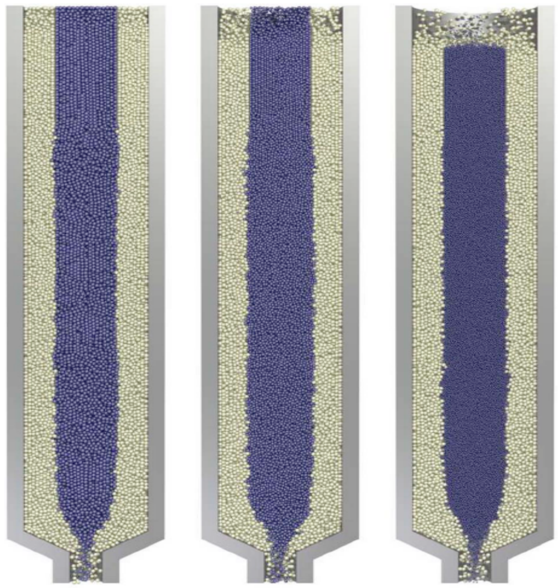
$$\forall i \in \mathcal{A}(\boldsymbol{q}^{(l)}, \epsilon)$$

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

Simulating the PBR nuclear reactor

- Problem of bidisperse granular flow with dense packing.
- Previous attempts: DEM methods on supercomputers at Sandia Labs regularization)
- 40 seconds of simulation for 440,000 pebbles needs 1 week on 64 processors dedicated cluster (Rycroft et al.)

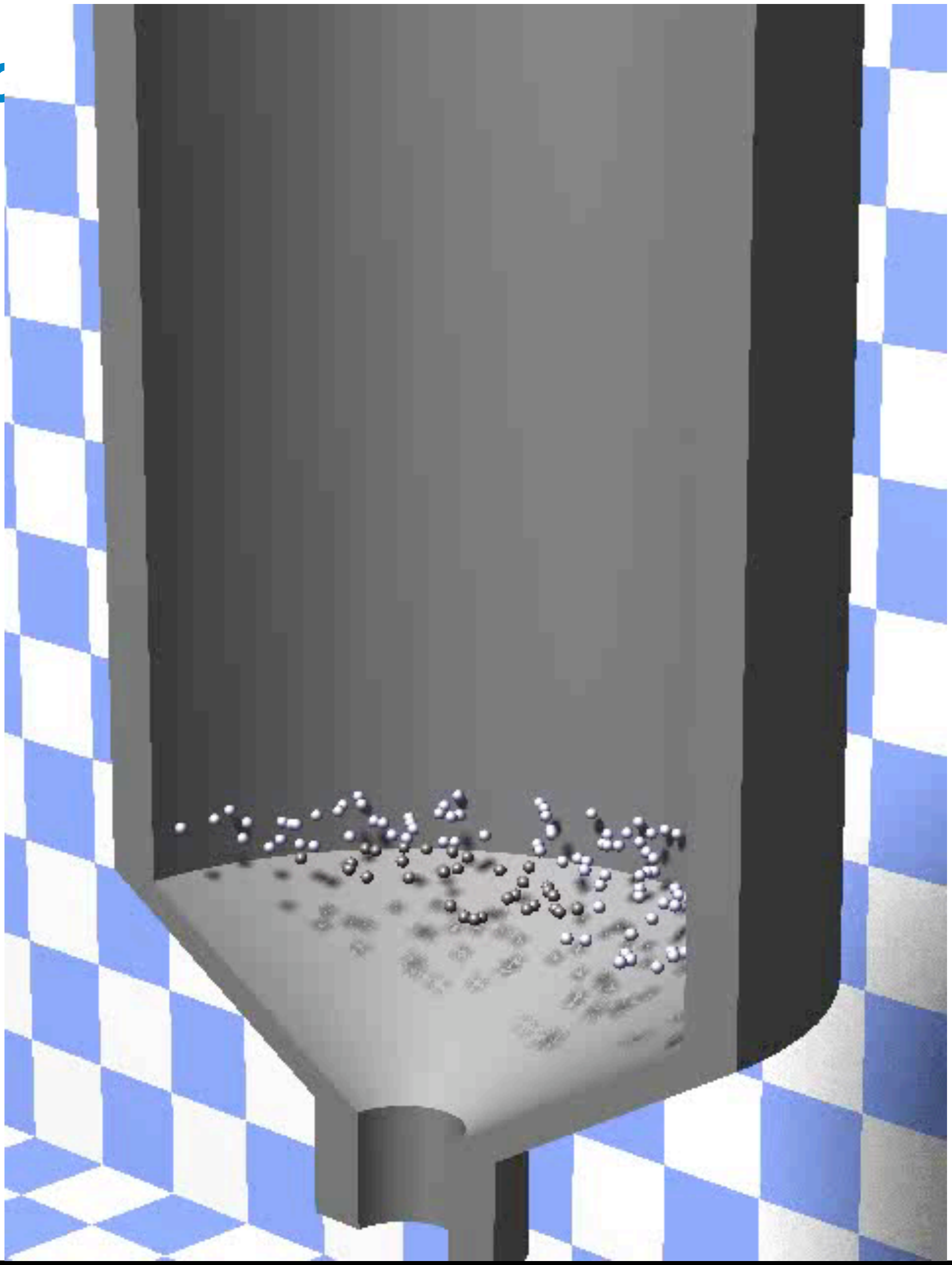

model a frictionless wall, $\mu_w=0.0$. For the current simulations we set $k_t=\frac{2}{7}k_n$ and choose $k_n=2\times10^5\ gm/d$. While this is significantly less than would be realistic for graphite pebbles, where we expect $k_n>10^{10}\ gm/d$, such a spring constant would be prohibitively computationally 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



Simulations with DEM. Bazant et al. (MIT and Sandia laboratories).

Simulating the PBR nuc

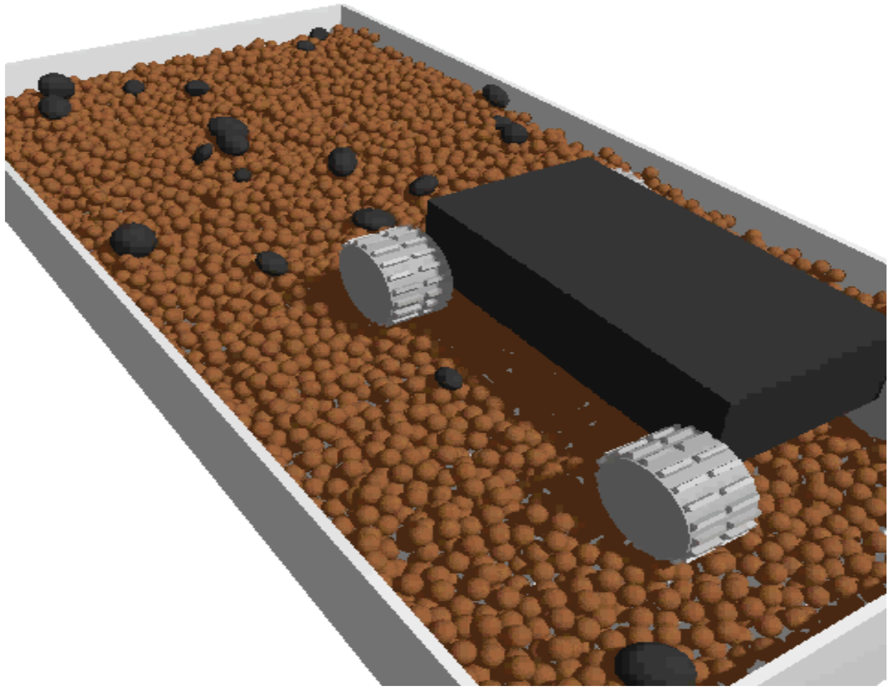
- 160’000 Uranium-Graphite spheres, 600’000 contacts on average
- Two millions of primal variables, six millions of dual variables
- *1 day on a Windows station…*
- But we are limited by the 2GB user mode limit, 64 bit port in progress—but linear scaling..
- We estimate 3CPU days, compare with 450 CPU days for an incomplete solution in 2006 !!!
- Answer 3: Our approach is efficient for large scale!!

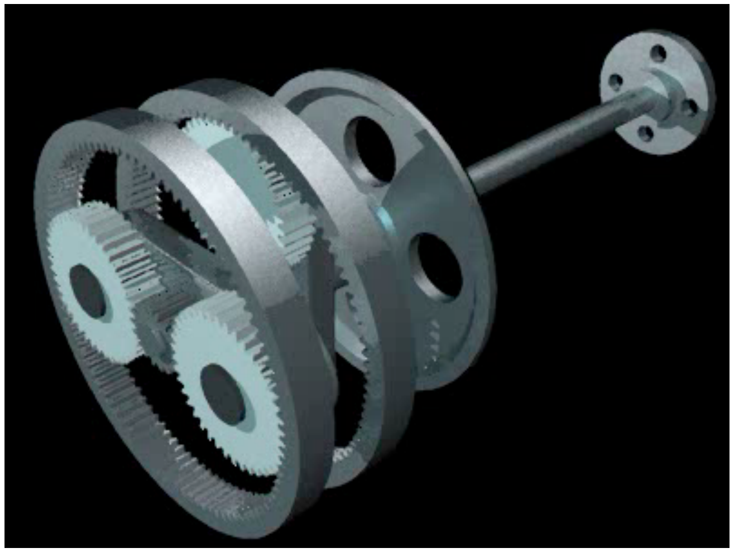


Research needs

- What do we need to extend the reach of DVI?

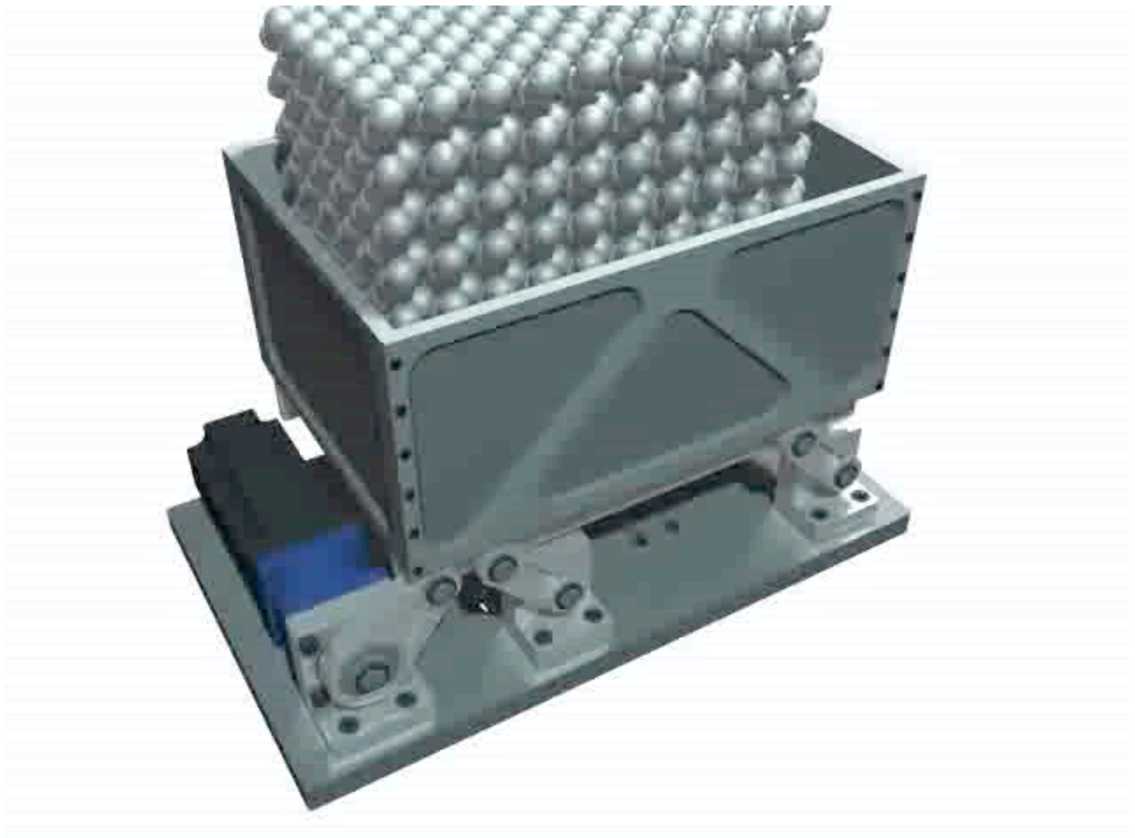
In addition, we can approach efficiently approach many engineering problems (see website for papers)





Scaling/constraint accuracy test:

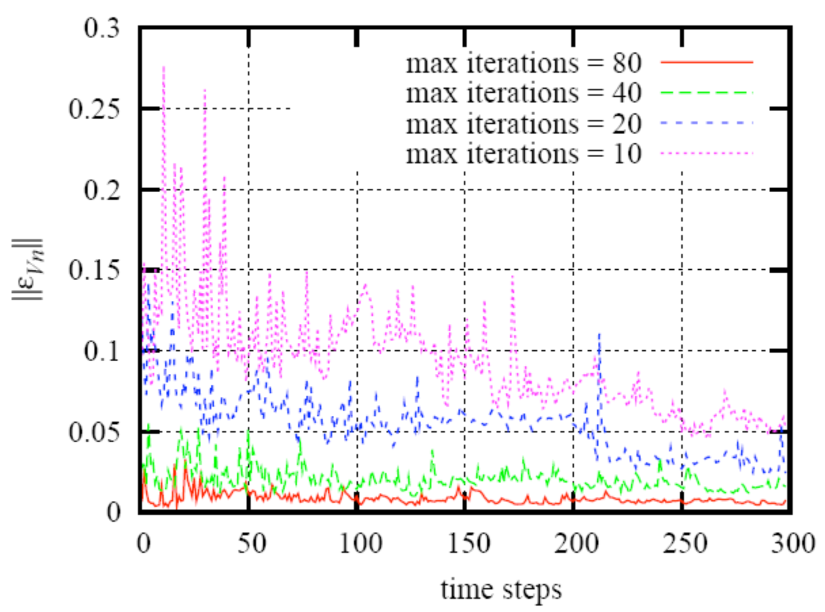
- Size-segregation in shaker, with thousands of steel spheres

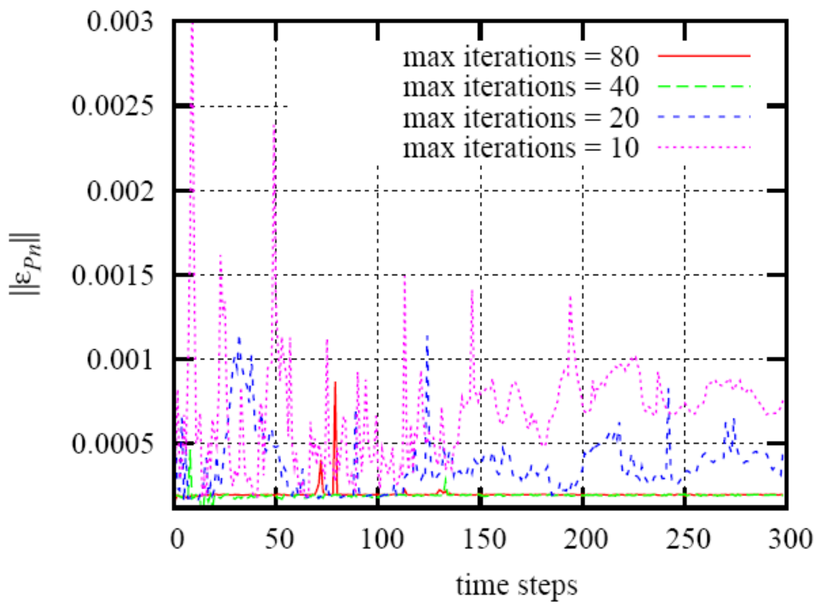


Note: solution beyond reach of Lemke-type LCP solvers!

Tests

- Feasibility accuracy increases with number of iterations:





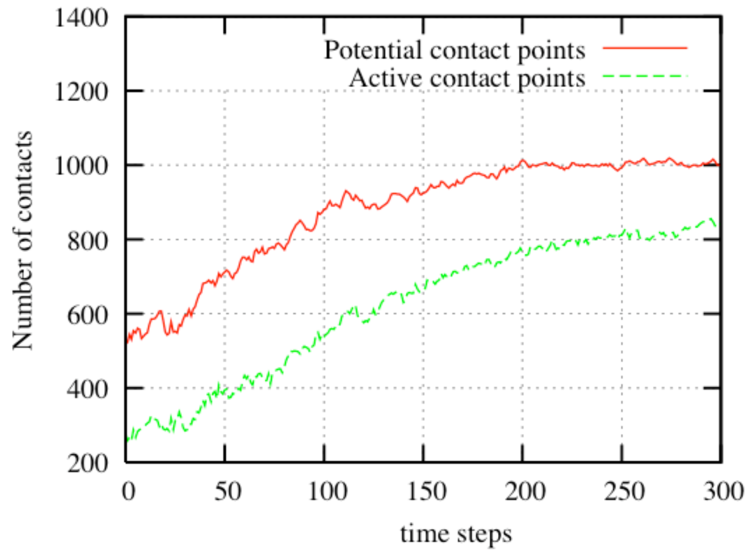
Speed violation in constraints

Position error in constraints (penetration)

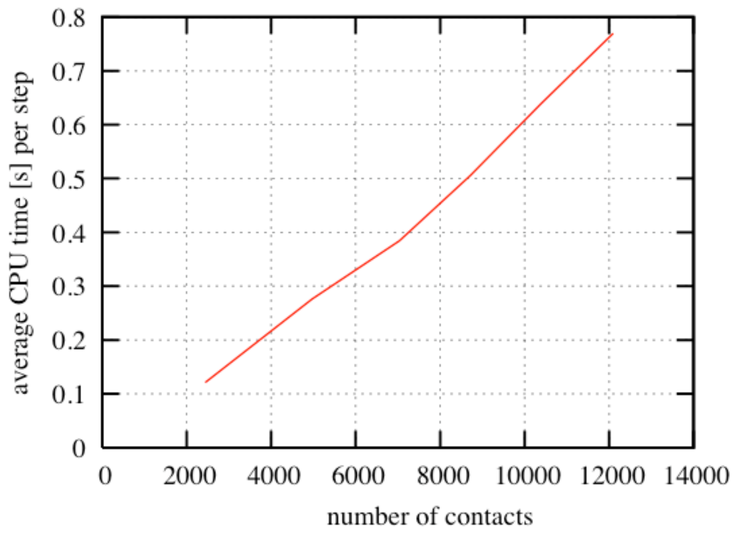
(with example of 300 spheres in shaker)

Tests: Scalability

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

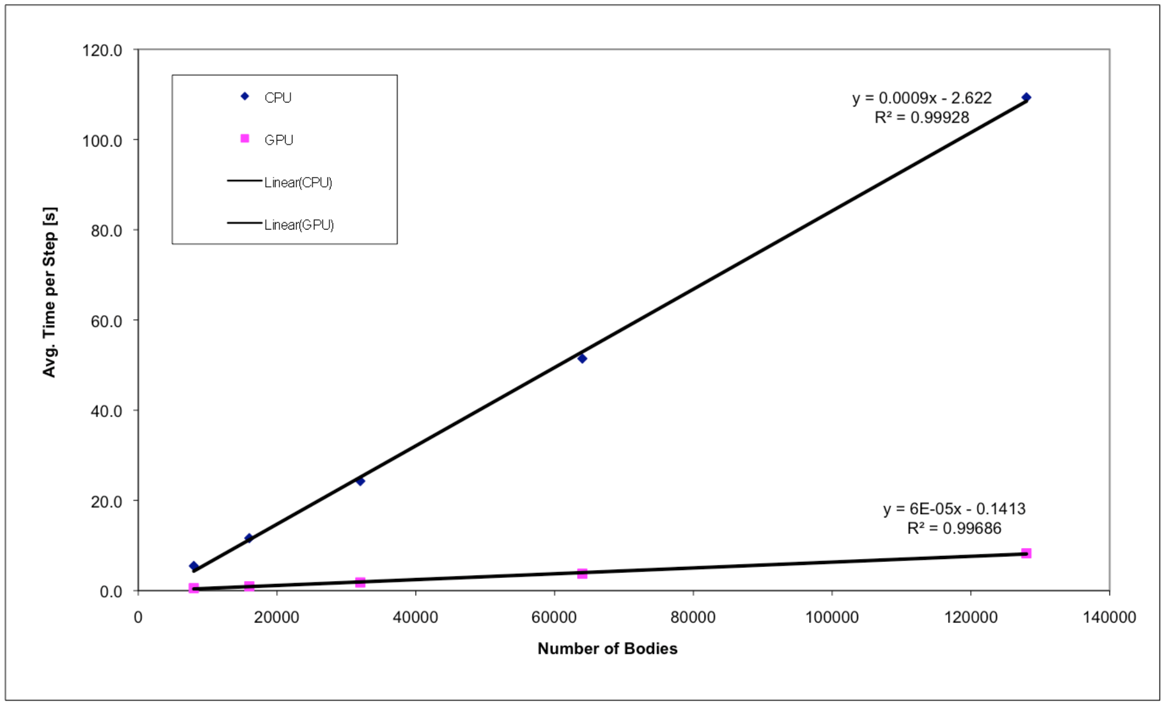


Number of contacts in time, 300 spheres



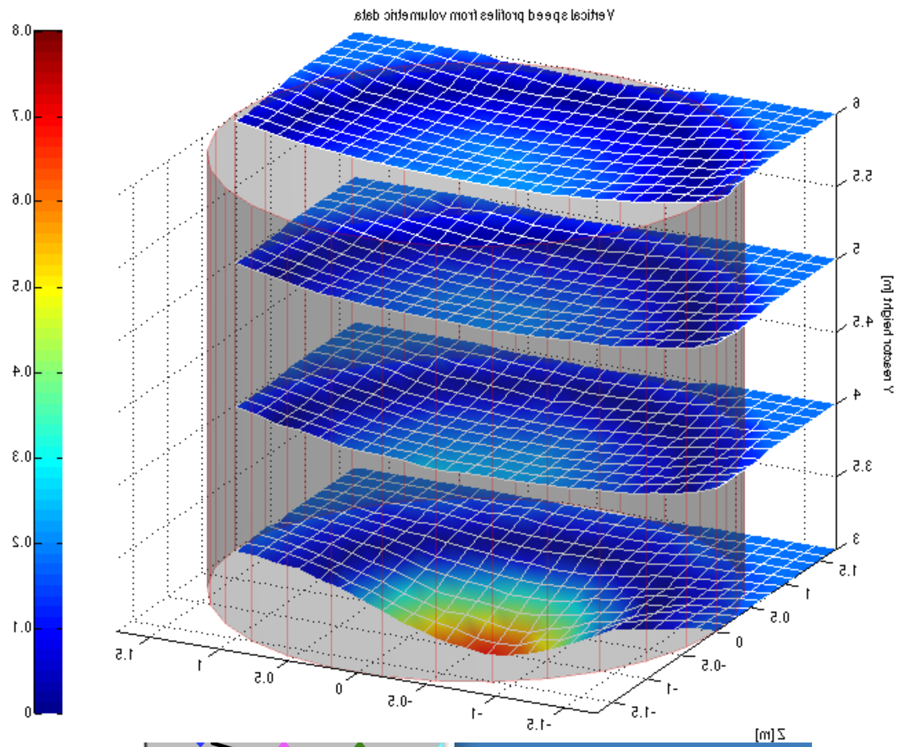
CPU time per step for 300-1500 spheres

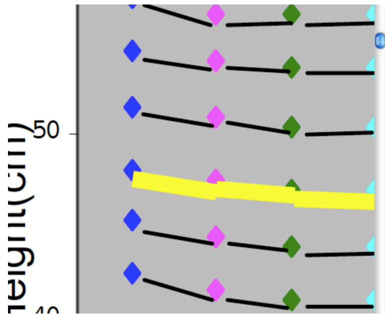
Preliminary Results for GS on large problems on GPU

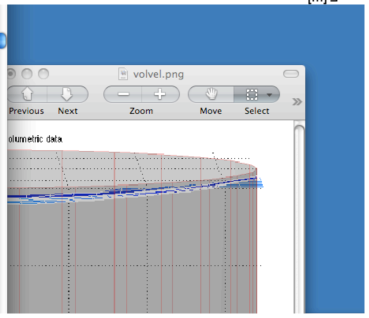


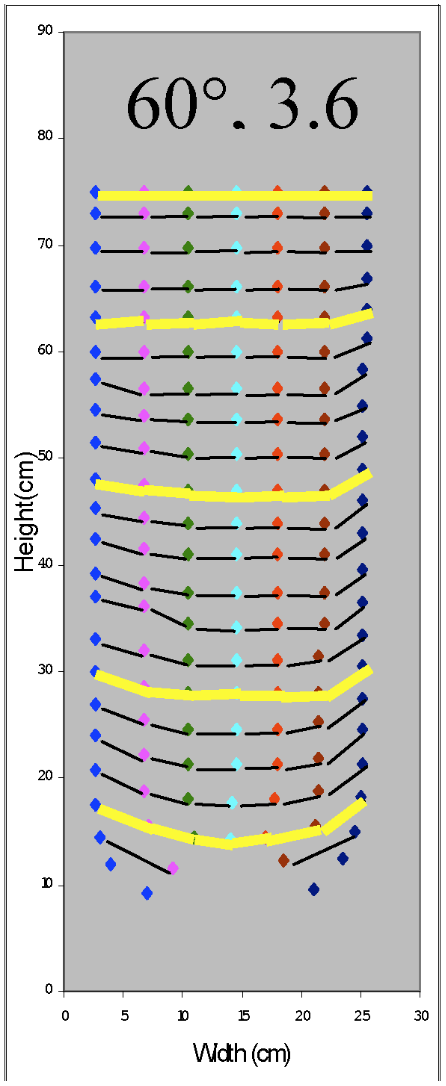
	CPU			GPU					
Number of Bodies	CCP Time	CD Time	Step Time	CCP Time	CD Time	Step Time	CCP Speedup	CD Speedup	Step Speedup
8000	5.26725	0.10655	5.47370	0.39600	0.12288	0.54658	13.3010	0.8671	10.0144
16000	11.18999	0.23601	11.63754	0.74096	0.15337	0.95315	15.1020	1.5388	12.2095
32000	23.28647	0.55861	24.28257	1.46889	0.21752	1.80878	15.8531	2.5680	13.4249
64000	49.16970	1.36559	51.44442	3.12925	0.36476	3.75152	15.7130	3.7438	13.7130
128000	103.66658	3.80176	109.34287	6.97682	0.74488	8.27050	14.8587	5.1038	13.2208

Comparison with experimental data PBR

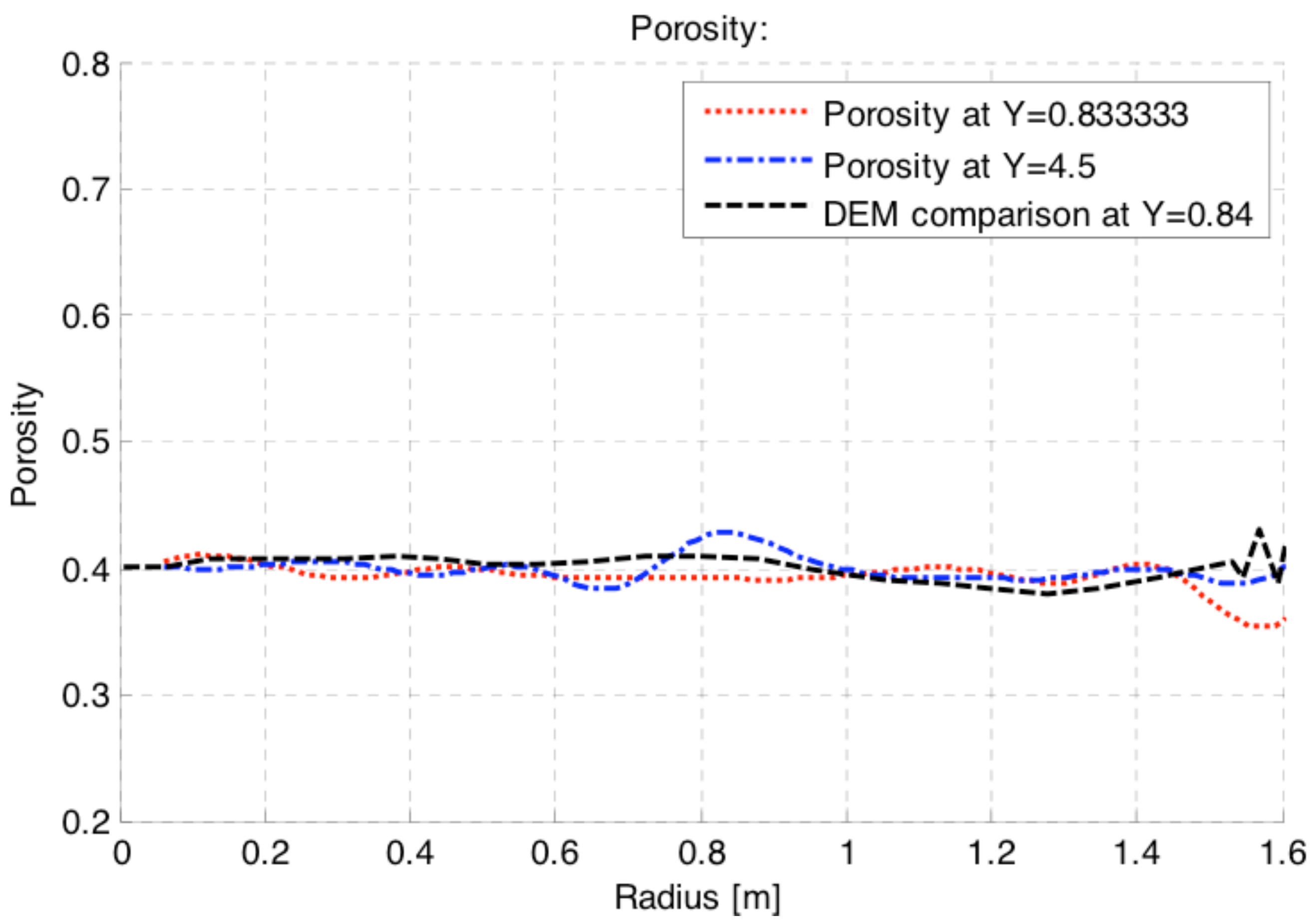








Packing statistics

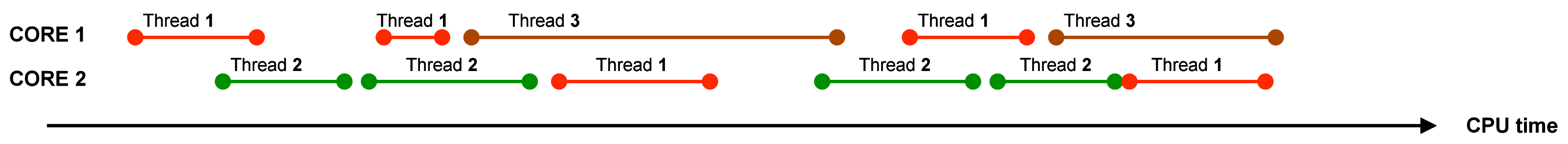


References (preprints are at authors' web site)

- M Anitescu, A. Tasora. ”An iterative approach for cone complementarity problems for nonsmooth dynamics”. Preprint ANL/MCS-P1413-0507, May 2007.
- M. Anitescu. Optimization-based simulation of nonsmooth dynamics. Mathematical Programming, series A, 105, pp 113–143, 2006.
- Madsen, J., Pechdimaljian, N., and Negrut, D., 2007. Penalty versus complementarity-based frictional contact of rigid bodies: A CPU time comparison. Preprint. TR-2007-05, Simulation-Based Engineering Lab, University of Wisconsin, Madison.

Multi-core multithreaded parallelism

- Trend in CPU: add multi-core. Today, dual cores. Tomorrow: 4,8,16, ..
- Multi-cores allow at least n_t threads (or *fibers*) to run in parallel:

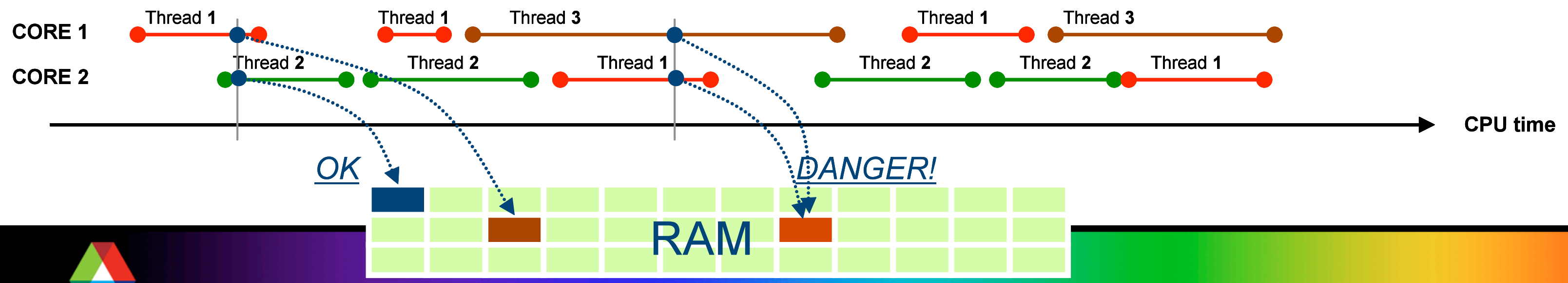


- *With n_c cores, maximum n_c**x** expected speedup (ex. <2**x** speedup in dual core)*

- In Win/Posix/Linux, multithreading C++ API is done in similar ways.
- Thread switching is almost instant (unlike *process* multitask switching)
- All RAM memory can be shared among threads

Multi-core multithreaded parallelism

- Problems caused by memory sharing in multithreading:
- Race conditions: since OS can switch threads ‘randomly’, the order of execution of parallel instructions can be *non-deterministic* → need of synchronization tricks (ex. *semaphores*)
- Resource locking: double write access to the same address of shared RAM must be avoided! → need of mutexes, or spinlocks, or semaphores



Multi-core multithreaded parallelism

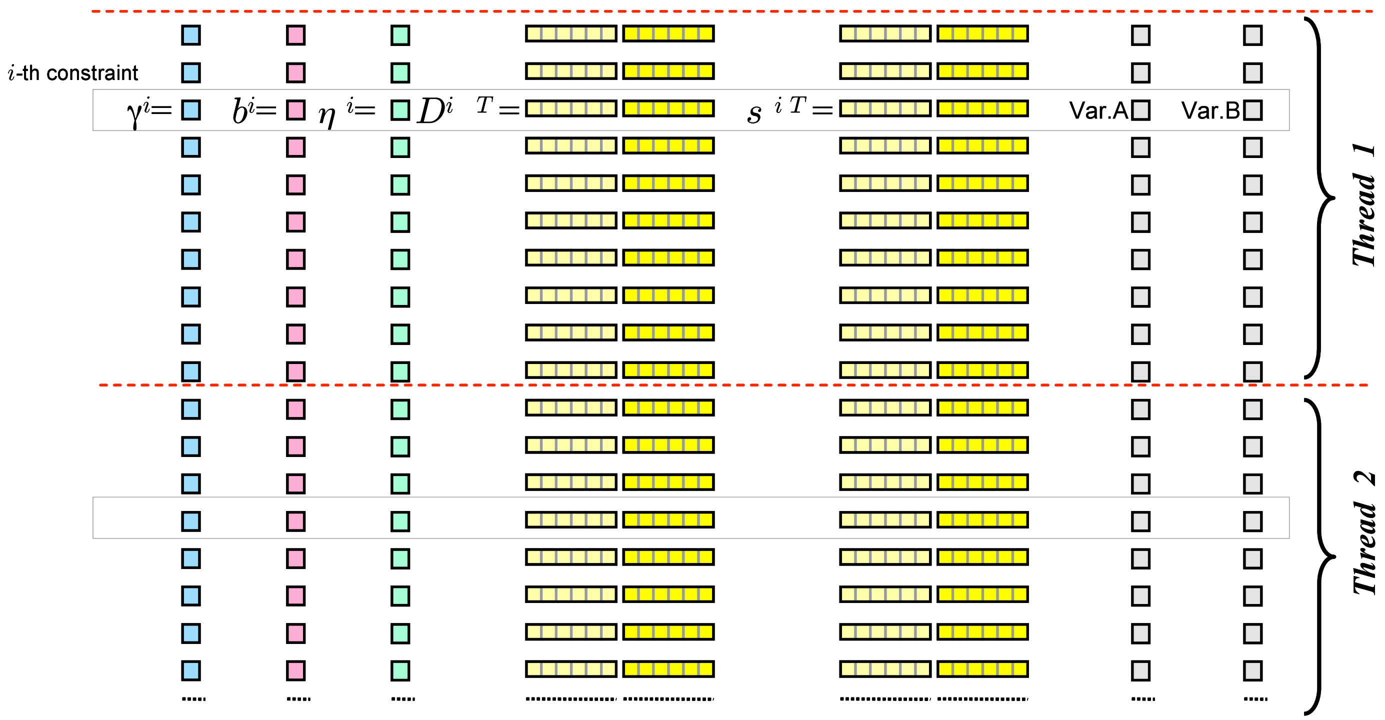
- A basic multi-threaded variant of our iterative solver can be easily implemented on multi-core CPU systems
- At each iteration, the loop over all constraints (the expensive part) is subdivided to multiple threads:
- Each update of constraint multiplier requires the update of the speed vector v (only for 6+6 elements relative to the two connected bodies), so there is the risk that two threads want to update the speed of the same body…
  - This 'conflict' risk is low if n_b **>>** n_t
  - Problem can be solved using spinlock or mutex
  - Nondeterministic algorithms? (no-sync)

Multi-core multithreaded parallelism

- Multithreaded version of the algorithm
- PHASE 1.a:

```
(1)   // Pre-compute some data for friction constraints
(2)   for $i := 1$ to $n_{\mathcal{A}}$
(3)       $\boldsymbol{s}_a^i = M^{-1} D^i$
(4)       $g_a^i = D^{i,T} \boldsymbol{s}_a^i$
(5)       $\eta_a^i = \frac{3}{\mathrm{Trace}(g_a^i)}$
```

- All contacts are divided among n_t threads
- Very simple: computations never overlap write addresses!

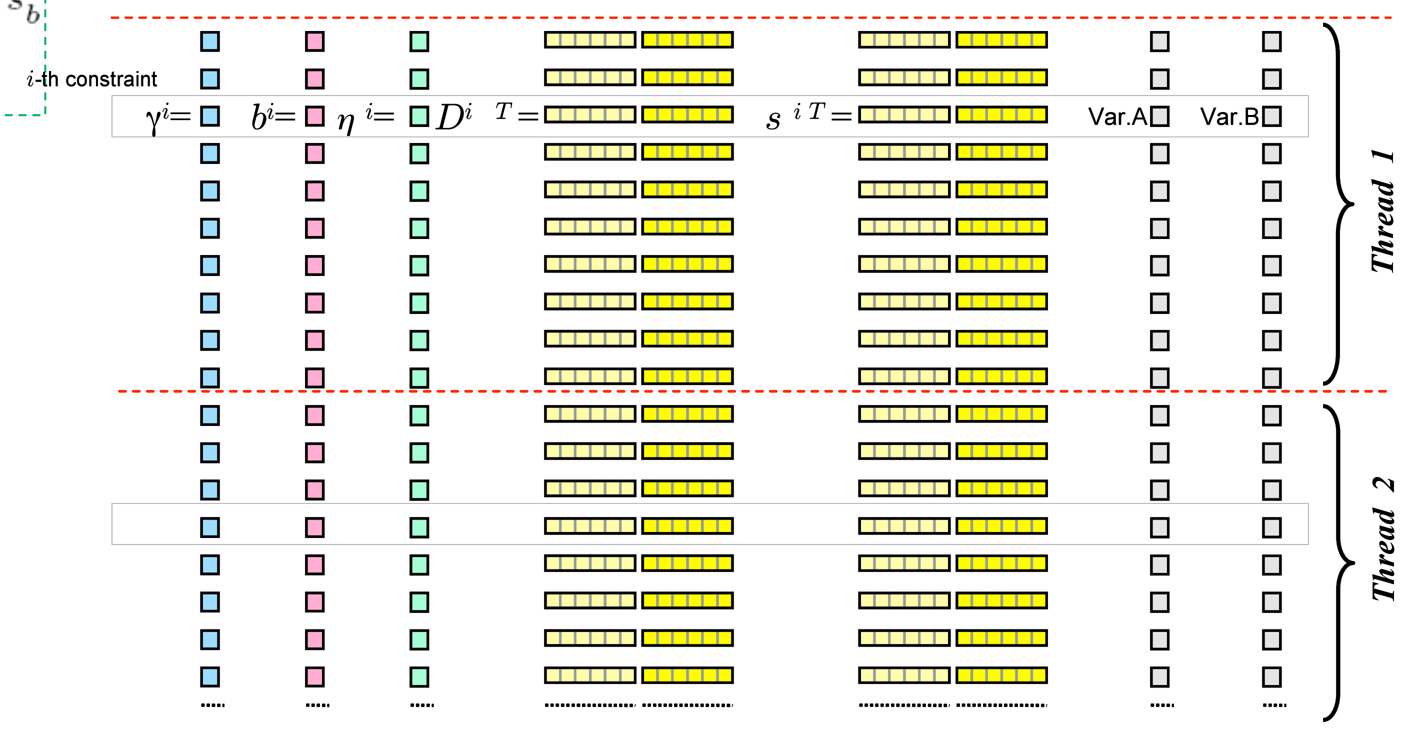


Multi-core multithreaded parallelism

PHASE 1.b:

```
(6)    // Pre-compute some data for bilateral constraints
(7)    for $i := 1$ to $n_{\mathcal{B}}$
(8)        $s_b^i = M^{-1}\nabla\Psi^i$
(9)        $g_b^i = \nabla\Psi^{i,T} s_b^i$
(10)       $\eta_b^i = \frac{1}{g_b^i}$
```

- All constraints are divided among n_t threads
- Very simple: computations never overlap write addresses!



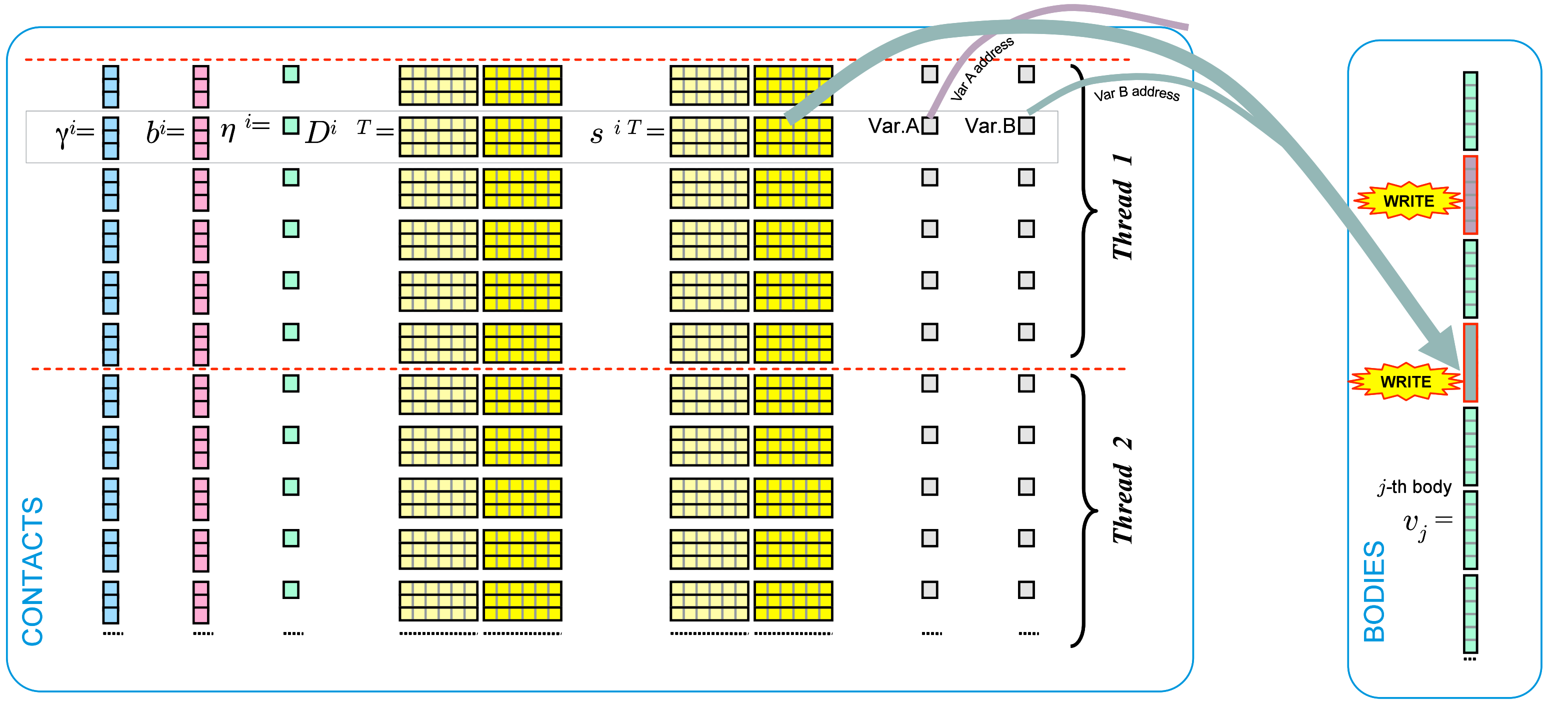
Multi-core multithreaded parallelism

PHASE 2.a: contacts

(18) // Initialize speeds

(19) $\boldsymbol{v} = \boxed{\sum_{i=1}^{n_\mathcal{A}} \boldsymbol{s}_a^i \gamma_a^{i,0}} + \sum_{i=1}^{n_\mathcal{B}} s_b^i \gamma_b^{i,0} + M^{-1}\tilde{\boldsymbol{k}}$

- All contacts are divided among n_t threads
- The results are 6+6 block-stores in speed vector, so there is the risk of WRITE CONFLICT!



Multi-core multithreaded parallelism

PHASE 2.b: bilateral

- The results are 6+6 block-increments in speed vector, so there is the risk of WRITE CONFLICT!

(18) // Initialize speeds

(19) $\boldsymbol{v} = \sum_{i=1}^{n_{\mathcal{A}}} \boldsymbol{s}_a^i \boldsymbol{\gamma}_a^{i,0} + \boxed{\sum_{i=1}^{n_{\mathcal{B}}} s_b^i \gamma_b^{i,0}} + M^{-1}\tilde{\boldsymbol{k}}$

[similar to PHASE 2.a – do not show graphs]

Multi-core multithreaded parallelism

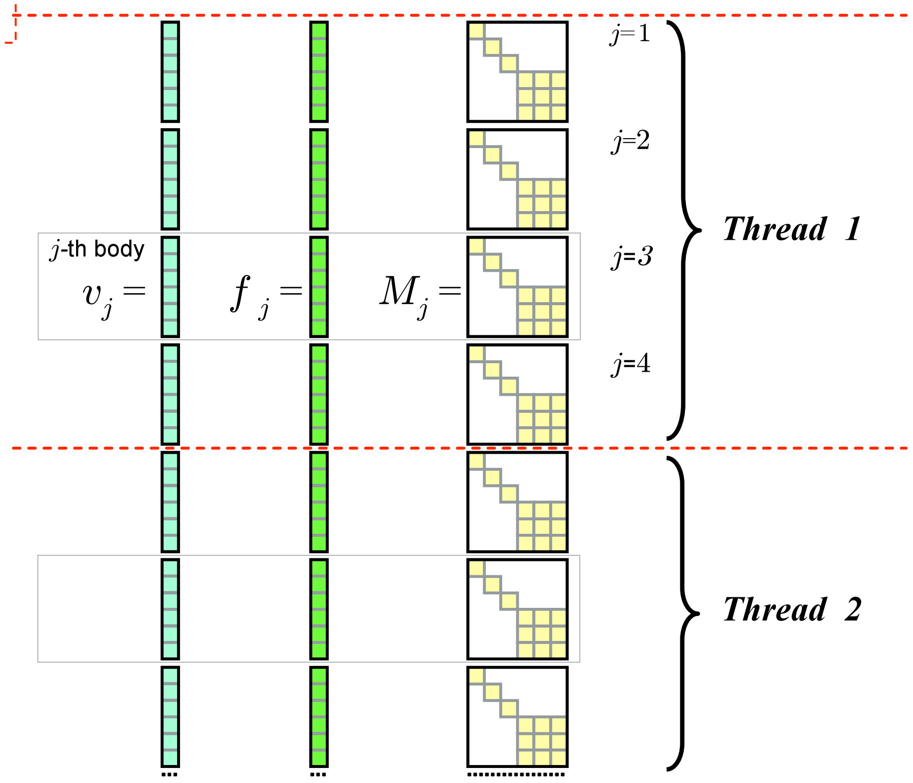
- All variables (bodies) are divided among n_t threads
- Very simple: computations never overlap write addresses!

■ PHASE 2.c:

$$// \text{ Initialize speeds}$$
$$\boldsymbol{v} = \sum_{i=1}^{n_\mathcal{A}} \boldsymbol{s}_a^i \gamma_a^{i,0} + \sum_{i=1}^{n_\mathcal{B}} s_b^i \gamma_b^{i,0} + \boxed{M^{-1}\tilde{\boldsymbol{k}}}$$

■ **Here just remember that**

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



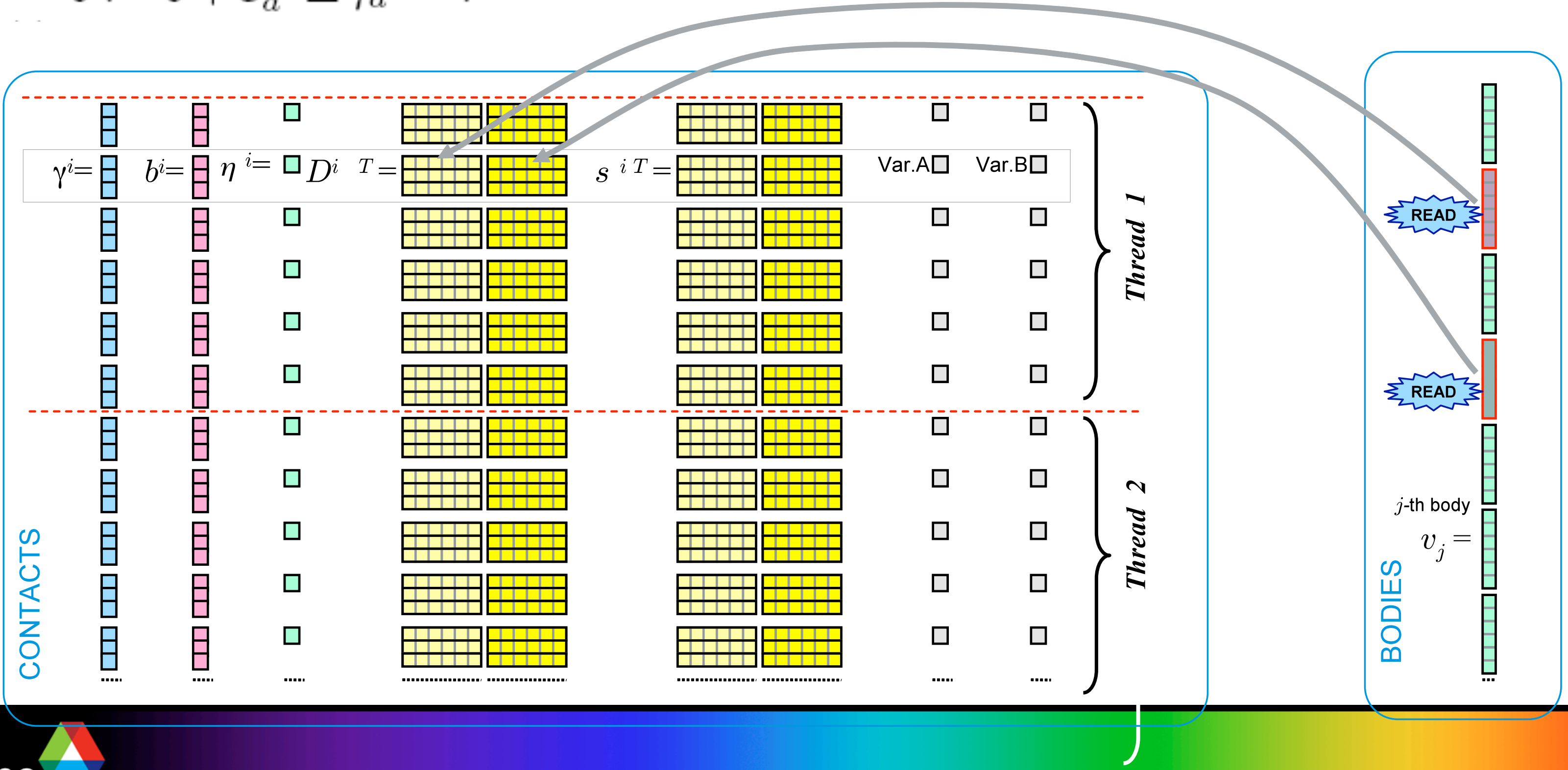
Multi-core multithreaded parallelism

- PHASE 3 (to be repeated for a set number of iterations):

```
(23)  // Loop on frictional constraints
(24)  for $i := 1$ to $n_{\mathcal{A}}$
(25)      $\boldsymbol{\delta}_a^{i,r} = \left(\boldsymbol{\gamma}_a^{i,r} - \omega\eta_a^i\left(D^{i,T}\boldsymbol{v}^r + \boldsymbol{b}_a^i\right)\right);$
(26)      $\boldsymbol{\gamma}_a^{i,r+1} = \lambda\Pi_\Upsilon\left(\boldsymbol{\delta}_a^{i,r}\right) + (1-\lambda)\boldsymbol{\gamma}_a^{i,r};$
(27)      $\Delta\boldsymbol{\gamma}_a^{i,r+1} = \boldsymbol{\gamma}_a^{i,r+1} - \boldsymbol{\gamma}_a^{i,r};$
(28)      $\boldsymbol{v} := \boldsymbol{v} + \boldsymbol{s}_a^{i^T}\Delta\boldsymbol{\gamma}_a^{i,r+1}.$
```

- All contacts are divided among n_t threads

This first sub-step is easy: computations never overlap write addresses



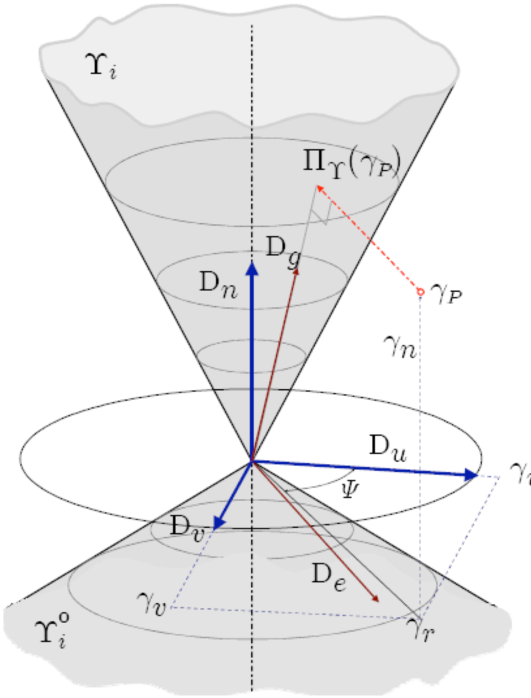
Multi-core multithreaded parallelism

- PHASE 3 (to be repeated for iterations):

```
3)  // Loop on frictional constraints
4)  for $i := 1$ to $n_{\mathcal{A}}$
5)      $\boldsymbol{\delta}_a^{i,r} = \left(\boldsymbol{\gamma}_a^{i,r} - \omega\eta_a^i \left(D^{i,T}\boldsymbol{v}^r + \boldsymbol{b}_a^i\right)\right);$
6)      $\boldsymbol{\gamma}_a^{i,r+1} = \lambda\Pi_{\Upsilon}\left(\boldsymbol{\delta}_a^{i,r}\right) + (1-\lambda)\boldsymbol{\gamma}_a^{i,r}\ ;$
7)      $\Delta\boldsymbol{\gamma}_a^{i,r+1} = \boldsymbol{\gamma}_a^{i,r+1} - \boldsymbol{\gamma}_a^{i,r}\ ;$
8)      $\boldsymbol{v} := \boldsymbol{v} + \boldsymbol{s}_a^{i^T}\Delta\boldsymbol{\gamma}_a^{i,r+1}.$
```

- All contacts are divided among n_t threads

This second sub-step is easy: it is the projection onto friction cone



Multi-core multithreaded parallelism

- PHASE 3 (to be repeated for iterations):

```
(23) // Loop on frictional constraints
(24) for $i := 1$ to $n_{\mathcal{A}}$
(25)     $\boldsymbol{\delta}_a^{i,r} = \left(\boldsymbol{\gamma}_a^{i,r} - \omega\eta_a^i \left(D^{i,T}\boldsymbol{v}^r + \boldsymbol{b}_a^i\right)\right);$
(26)     $\boldsymbol{\gamma}_a^{i,r+1} = \lambda\Pi_{\Upsilon}\left(\boldsymbol{\delta}_a^{i,r}\right) + (1-\lambda)\boldsymbol{\gamma}_a^{i,r};$
(27)     $\Delta\boldsymbol{\gamma}_a^{i,r+1} = \boxed{\boldsymbol{\gamma}_a^{i,r+1} - \boldsymbol{\gamma}_a^{i,r};}$
(28)     $\boldsymbol{v} := \boldsymbol{v} + \boldsymbol{s}_a^{i^T}\Delta\boldsymbol{\gamma}_a^{i,r+1}.$
```

This third sub-step is easy

- All contacts are divided among n_t threads

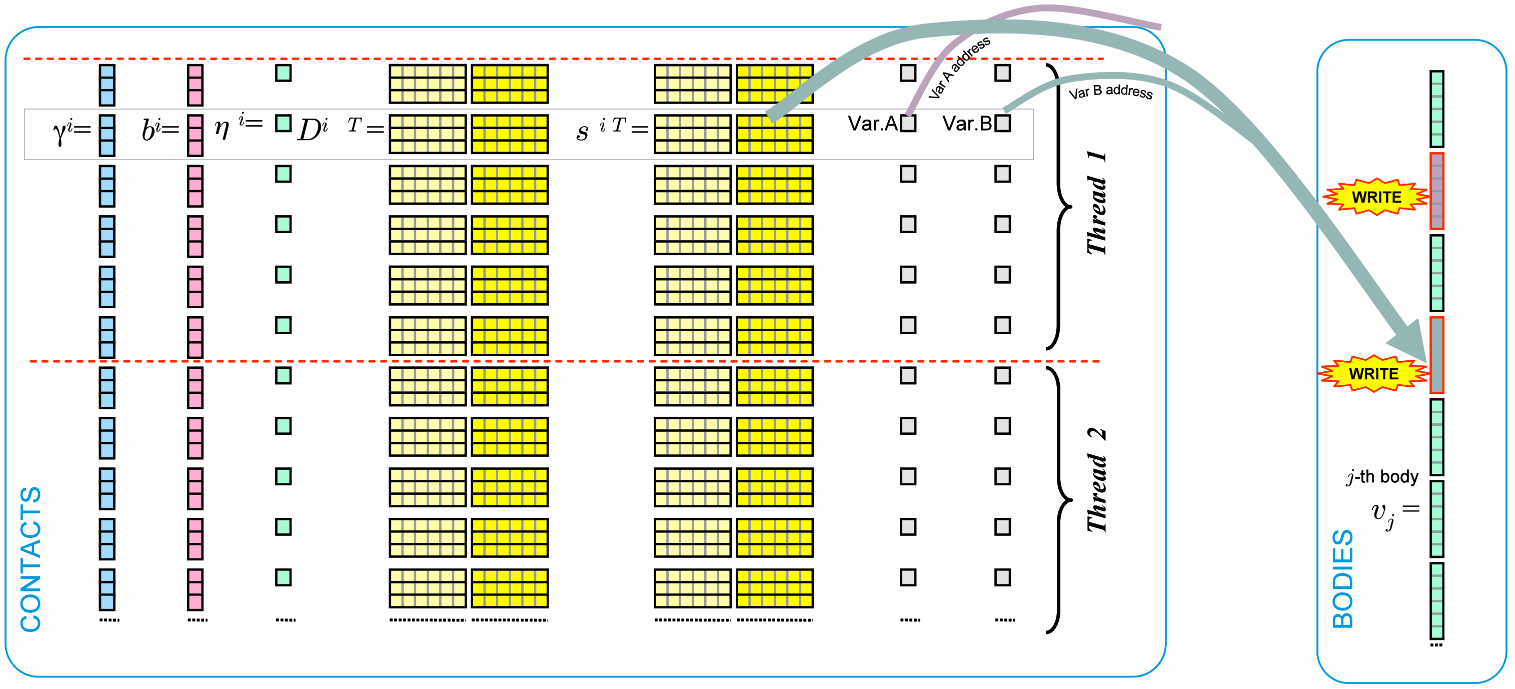
Multi-core multithreaded parallelism

- PHASE 3 (to be repeated for iterations):

```
// Loop on frictional constraints
for $i := 1$ to $n_{\mathcal{A}}$
    $\boldsymbol{\delta}_a^{i,r} = \left(\boldsymbol{\gamma}_a^{i,r} - \omega\eta_a^i\left(D^{i,T}\boldsymbol{v}^r + \boldsymbol{b}_a^i\right)\right);$
    $\boldsymbol{\gamma}_a^{i,r+1} = \lambda\Pi_{\Upsilon}\left(\boldsymbol{\delta}_a^{i,r}\right) + (1-\lambda)\boldsymbol{\gamma}_a^{i,r}\;;$
    $\Delta\boldsymbol{\gamma}_a^{i,r+1} = \boldsymbol{\gamma}_a^{i,r+1} - \boldsymbol{\gamma}_a^{i,r}\;;$
    $\boldsymbol{v} := \boldsymbol{v} + \boldsymbol{s}_a^{i^T}\Delta\boldsymbol{\gamma}_a^{i,r+1}.$
```

- All contacts are divided among n_t threads

***This fourth sub-step is similar to phases 2.a/2.b: results are 6+6 block-stores in speed vector, so there is the risk of* WRITE CONFLICT!**



Multi-core multithreaded parallelism

- PHASE 3 (to be repeated for iterations):

```
(29)    // Loop on bilateral constraints
(30)    for $i := 1$ to $n_{\mathcal{B}}$
(31)        $\delta_b^{i,r} = \left(\gamma_b^{i,r} - \omega\eta_b^i\left(\nabla\boldsymbol{\Psi}^{i,T}\boldsymbol{v}^r + b_b^i\right)\right);$
(32)        $\gamma_b^{i,r+1} = \lambda\Pi_{\Upsilon}\left(\delta_b^{i,r}\right) + (1-\lambda)\gamma_b^{i,r}$ ;
(33)        $\Delta\gamma_b^{i,r+1} = \gamma_b^{i,r+1} - \gamma_b^{i,r}$ ;
(34)        $\boldsymbol{v} := \boldsymbol{v} + s_b^{i^T}\Delta\gamma_b^{i,r+1}.$
```

- All constraints are divided among n_t threads

(we do not discuss the loop on bilateral constraints because it is the same as the four steps detailed before)

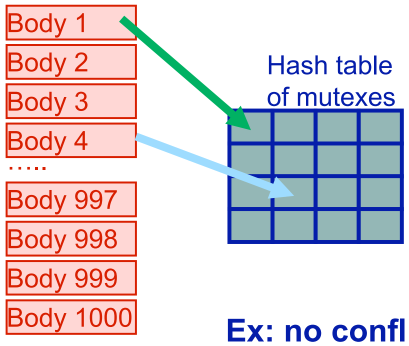
Multi-core multithreaded parallelism

- How to avoid the possible write-conflicts in shared memory, introduced in Phases 2.a, 2.b and 3 (fourth step) ?
- SOLUTION 0
- Do nothing, and just tolerate the risk of non-atomic updates of floating point values.
(Maybe I've been lucky: this never caused problems so far… Anyway, remember Murphy's laws, etc.)
- SOLUTION 1
- Introduce a shared single mutex (called 'critical section' in Windows API) to be raised each time the vector is written by some thread.
- + simple implementation
-- each single thread will halts all threads even when not needed
- SOLUTION 2
- Allocate n_b mutexes, each per body.
- + simple implementation, good performance
- -- will waste RAM when dealing with thousands of bodies

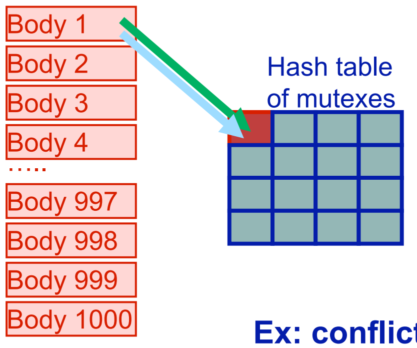
Multi-core multithreaded parallelism

- How to avoid the possible write-conflicts in shared memory, introduced in Phases 2.a, 2.b and 3 (fourth step) ?
- SOLUTION 3
- Allocate a fixed amount of mutexes into an hash-table. When a thread need to write in a body, it quickly transforms the body index into an hash-table index using an hash function, then blocks the corresponding mutex into that bucket.
- ***If the number of mutexes in the hash table is an order of magnitude larger than the number of parallel threads, it is unlikely that we get 'false conflict', and performance is as good as if we have as many mutexes as bodies.***
- + good performance, do not waste RAM
- -- not so easy to implement

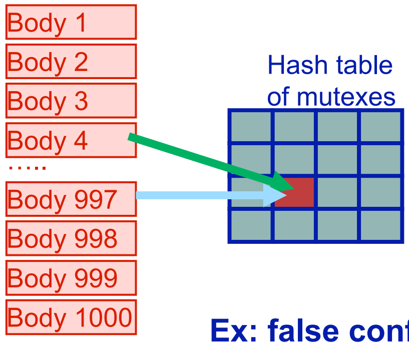
→ Hashing fx (thread 1)
→ Hashing fx (thread 2)



Ex: no conflict



Ex: conflict
(one thread will wait for the other)



Ex: false conflict
(one thread will wait for the other)

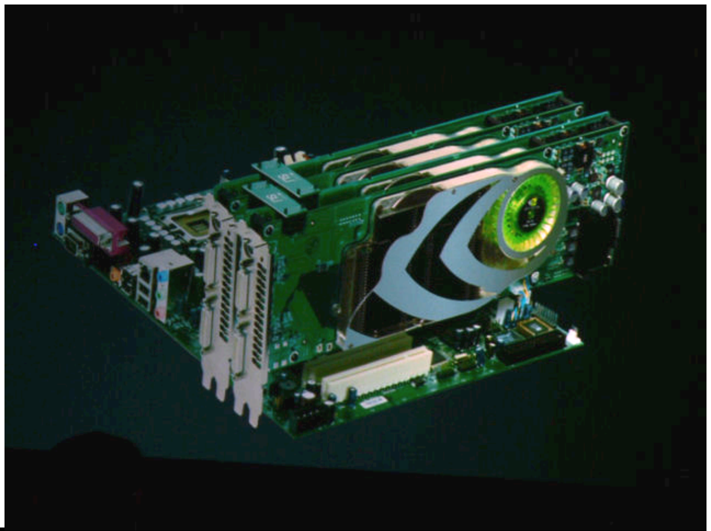
GP-GPU stream-kernel parallelism

- GPU, Graphical Processor Units = many “stream processors”
- Clusters of “multiprocessors” (→ hundreds of parallel threads)
- Recent GPU have floating-point capability claimed IEEE single

  → Can be used for general purpose parallel computation (GP-GPU = General Purpose GPU)

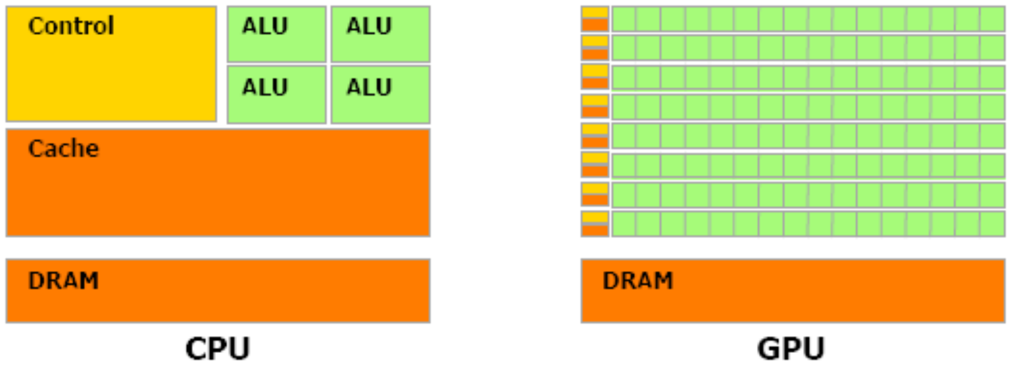
  → Nvidia offers CUDA as a programming API-SDK to exploit GPU multiprocessors for scientific computing! (G80 family)

  → Nvidia sells cheap TESLA boards, a low-cost way to get 1TeraFlop parallel computing power.



GP-GPU stream-kernel parallelism

- Performance: > 1000 GFLOPS with recent GPU processors



GP-GPU stream-kernel parallelism

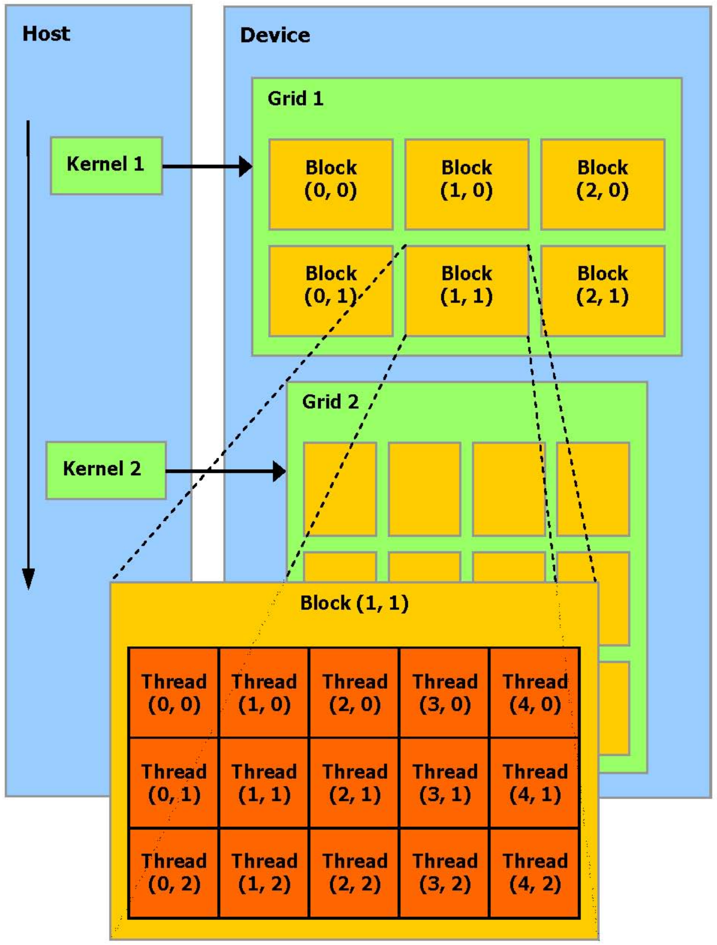
- SIMD **(non-VonNeumann)** architecture:
- Single Instruction (“*kernel*”) on..
- Multiple Data (“*stream*”)

- Our iterative method for multibody can be implemented as SIMD

- High internal memory bandwidth (>80 GB/s),
  but CPU ←→GPU bottleneck (max 3.7 GB/s upload, 2 GB/s download)

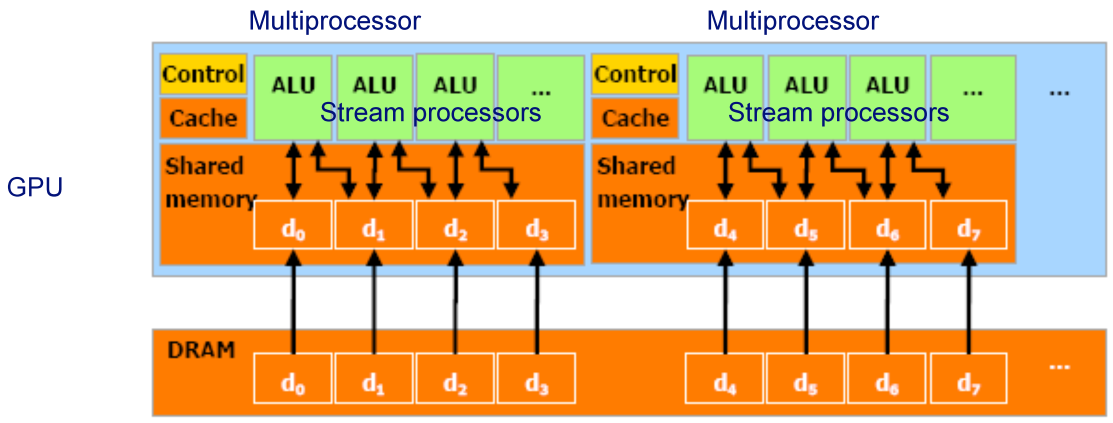
GP-GPU stream-kernel parallelism

- We adopt the CUDA SDK to develop C++ code which can easily exploit the capabilities of the G80 boards from NVIDIA.
  - **With CUDA, one can write '*kernels*' (functions executed in parallel) in C-like language.**
  - **Kernels are executed in '*blocks*' of parallel threads.**
  - **Multiple blocks can be arranged in '*grids*'**



GP-GPU stream-kernel parallelism

- Memory issues..
- *Not all memory write/reads are cost-less*
- On G80 boards :
- Global memory: >1Gb of DRAM (but >100 clock cycles of write/read latency)
- Shared memory: few kBytes per multiprocessor, but no latency



- *Shared memory has limitations:*
- A shared memory cache *per thread block* → blocks cannot communicate
- Shared memory cache is *limited in size* (16kBytes on G80 boards)

GP-GPU stream-kernel parallelism

- Relevant design constraints
  - How many **threads per thread block**? Max 512 (hardware limit)

    Too many threads: waste the shared memory cache.

    Too few threads: multiprocessors cannot hide global memory latency.

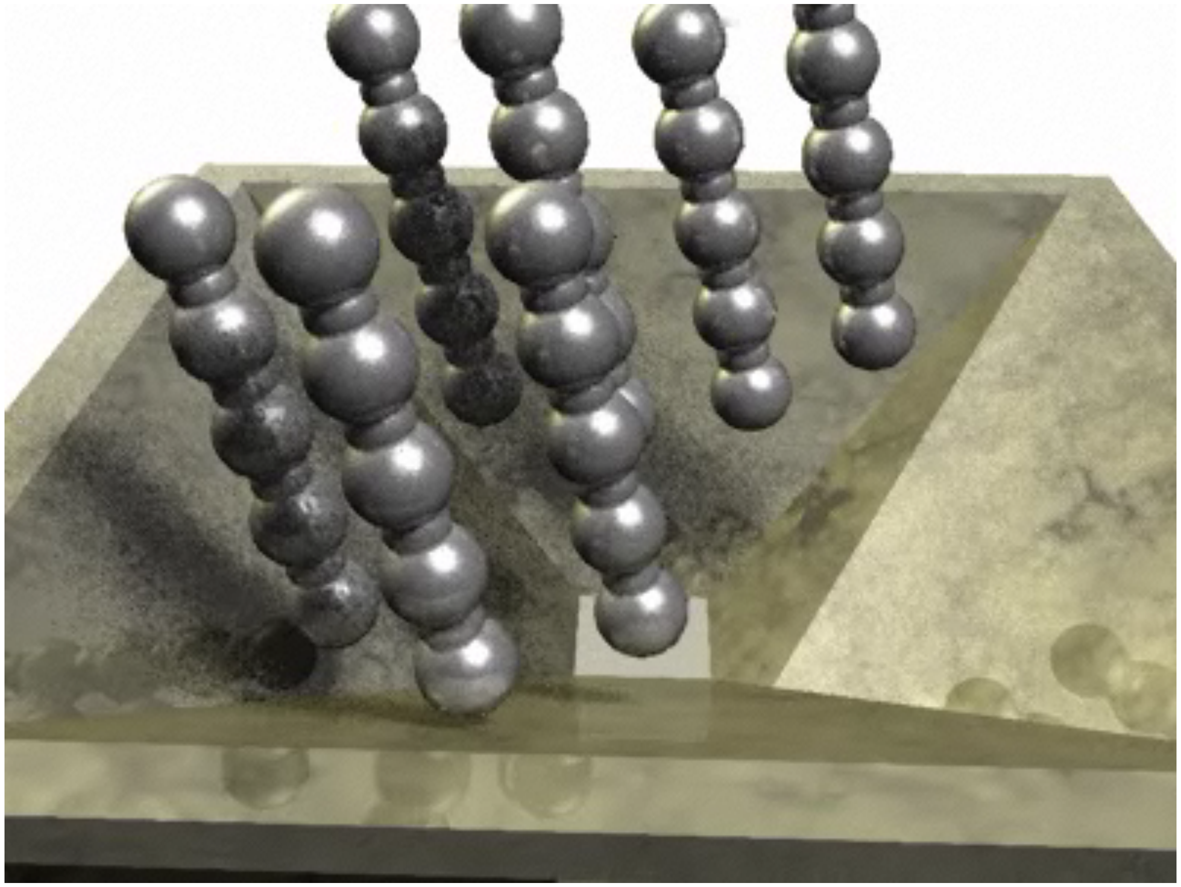
    Suggested: 128 threads per block or more.
  - How many **thread blocks per grid**?

    At least as many as available multiprocessors (ex. 12 on 8800 GT)

    Suggested: 100 blocks per grid or more.

Examples

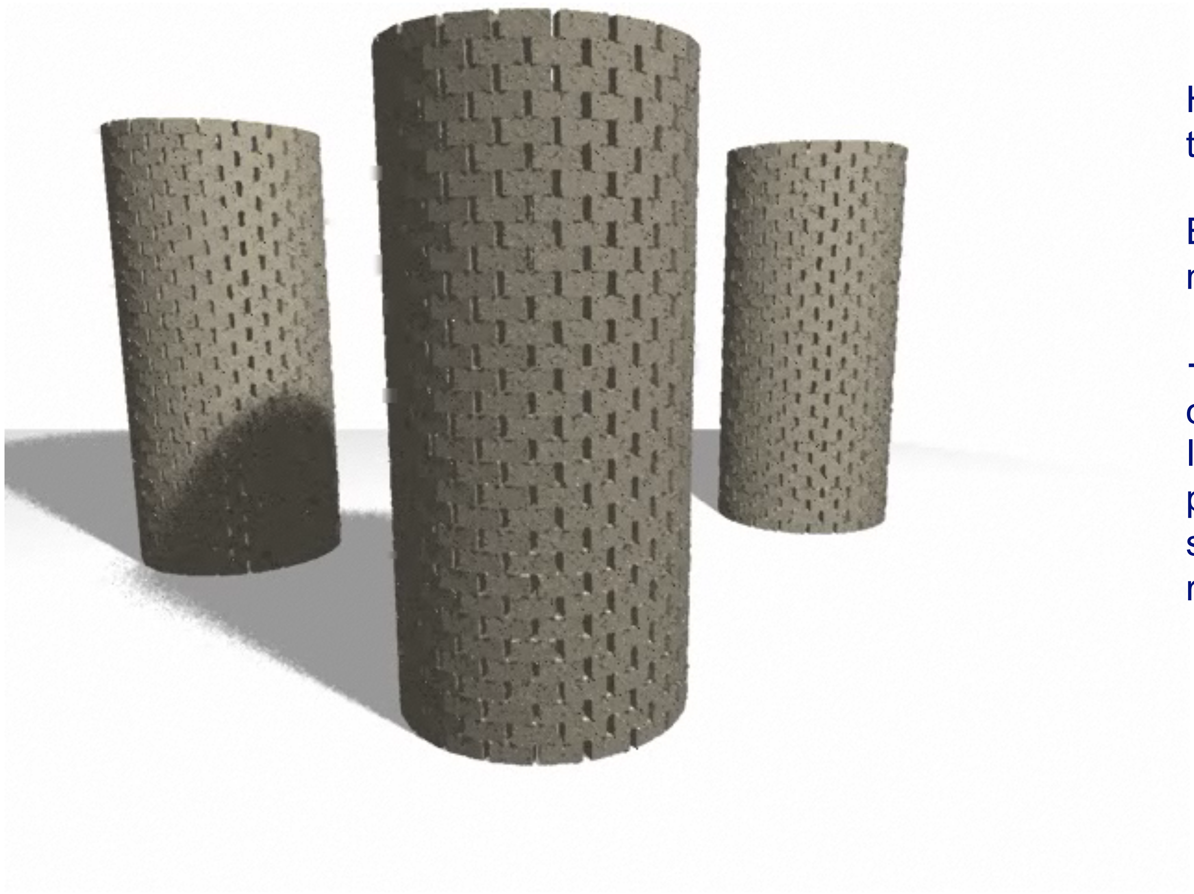
- Benchmark: contacts between articulated objects (contacts + bilateral joints)



Note: real-time simulation

Examples

- Benchmark: contacts between many polytopes and high friction coeff.



High stacks of objects: a typical difficult benchmark.

Even smallest CCP errors may compromise stability.

→ Many iterations needed, otherwise unstable and falls. Indicates need for preconditioning or warm starting; subject of future research.

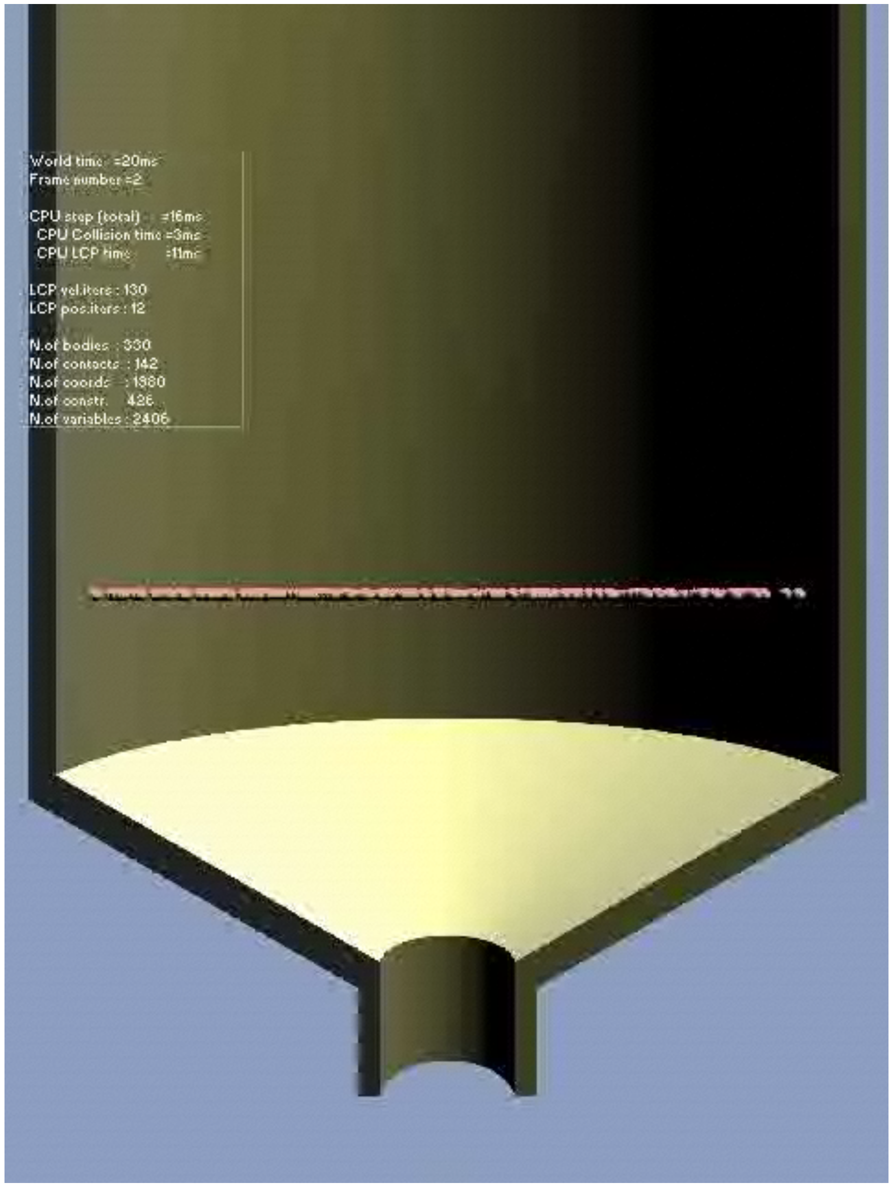
100 iterations: STABLE

Few iterations: UNSTABLE

Examples

- Refueling in a PBR nuclear reactor
  - Up to 300’000 uranium-graphite pebbles
  - Dense bidisperse granular flow problem: hard to simulate
  - Our method requires hours on 1CPU where other approach (DEM) requires weeks on 64 .
  - Currently, we are doing 30,000 pebbles in about 2 hrs CPU on windows Laptop, maximum 120,000 contacts and +500,000 impulse variables.

For details on PBR reactors, see Bazant et al. (MIT)



```
World time   =20ms
Frame number =2

CPU step (total)    =16ms
  CPU Collision time =3ms
  CPU LCP time       =11ms

LCP vel.iters : 130
LCP pos.iters : 12

N.of bodies   : 330
N.of contacts : 142
N.of coords   : 1980
N.of constr.    426
N.of variables : 2406
```

Some open problems

- Slow convergence for high mass ratios (many iterations needed for reasonable precisions).
- How to introduce a more advanced friction model? Ex: different static and kinematic friction coefficients, etc.
  Note: anisotropic friction would be easy.
- How to precondition (to make the number of iterations constant for same penetration error and vastly increasing number of bodies?) though warm starting may contribute a lot here.
- Collisions with restitution: not discussed here..

Future work

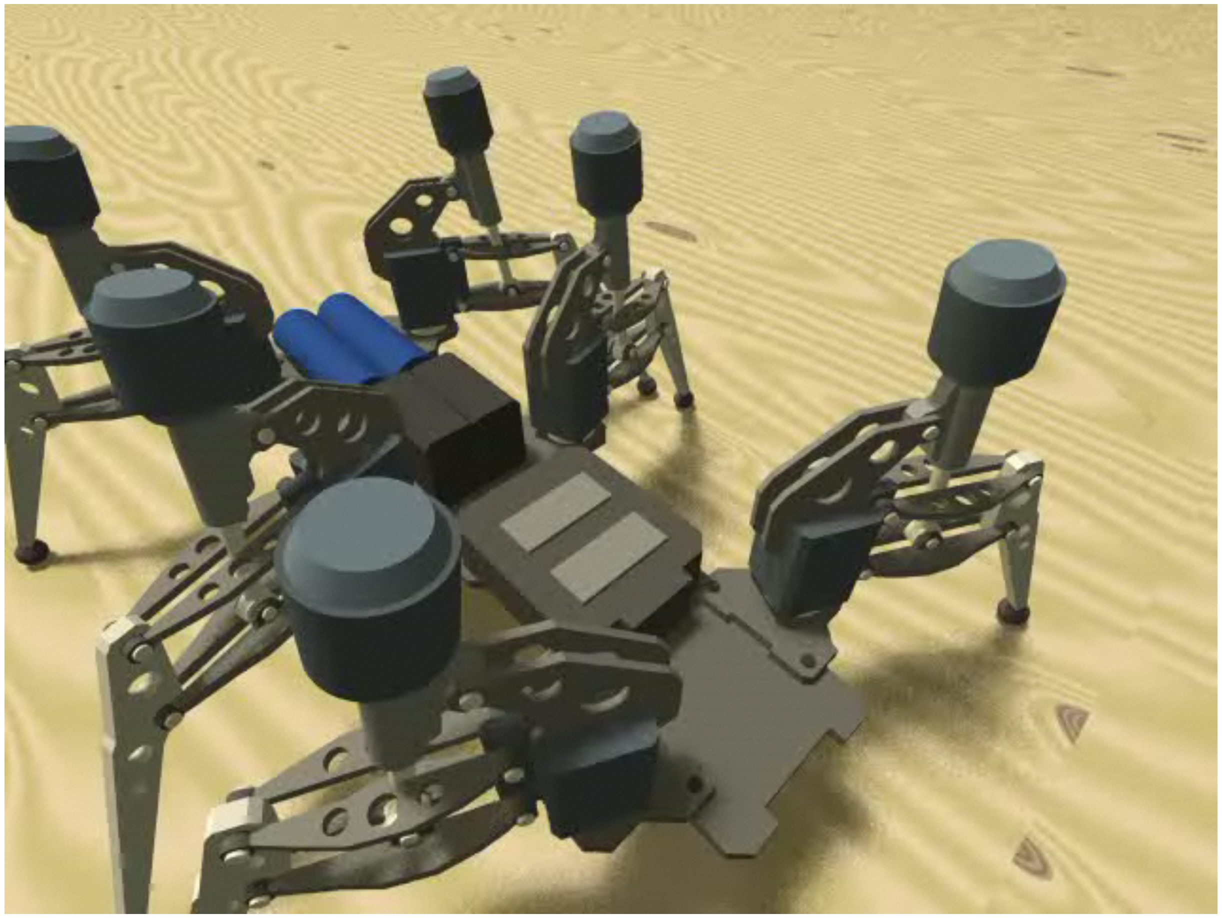
- Optimization of C++ source, aiming at simulations with 500’000 bodies on a single PC.
- Implementation on parallel machines. Multithreading.
- Persistent contact manifold, for warm starting the method
- Recent GP-GPU processors should be exploited for massive low-cost parallelism (SIMD stream-kernel architecture)

Conclusions

- Approach based on Time Stepping Methods
- Solution by means of fixed point iteration, with no polygonal cone approximation.
- Fast, robust, matrix-less iterative scheme
- Fits well in real-time scenarios, since can be stopped early with a good solution.
- Tested with more than 500’000 constraint multipliers, scales well.
- Implemented in a C++ multibody simulation middleware Chrono::Engine (developer by Tasora) `(http://www.deltaknowledge.com/chronoengine)`

Introduction

- Example: a walking robot, simulated using our approach



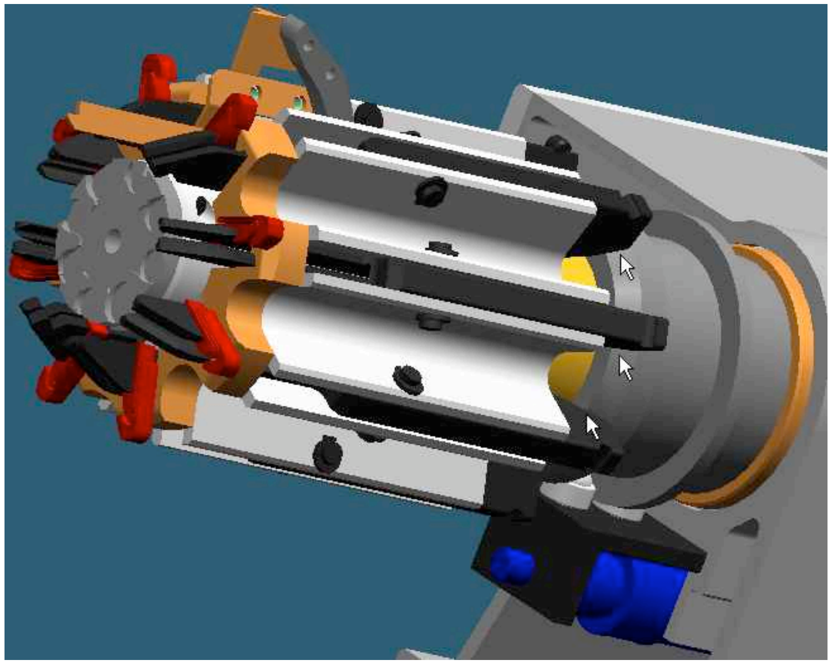
Introduction

- Unilateral constraints, friction, impacts are frequent in mechanisms
- They result in discontinuities in velocity→ non-smooth dynamics

Traditional ODE / DAE solvers require **smoothness**!

Example: packaging device

Example: radial gripper with 3D cams

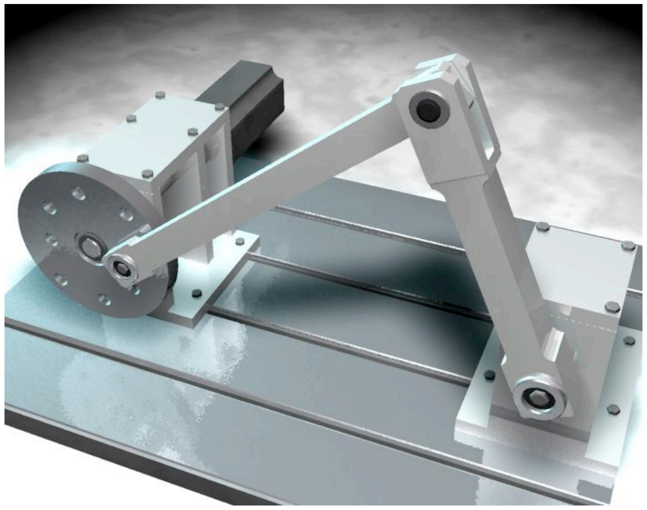


Introduction

- Multi rigid body systems, n degrees of freedom

Only bilateral constraints:

DAE / ODE:
Solve for unknown accelerations at each time step using **linear systems**



O(n^3) **computational complexity** [Gauss]

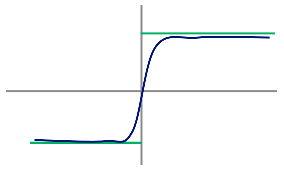
O(n) **computational complexity** with recent methods [A.Tasora, D.Baraff]

Introduction

- **.. Adding also non-smooth constraints (es.friction):**

ODE or DAE + regularization methods
(trick: approximate with stiff force fields)

N O



Stiffness: too small time steps!

ODE or DAE + "stop-and-restart" methods

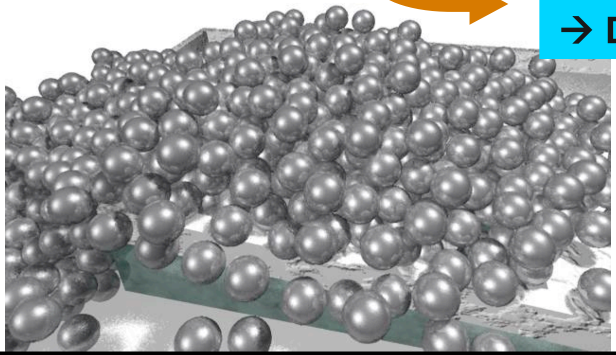
N O

Impracticable for complex systems!

"Time Stepping Methods" ,
"Vector Measure Differential Inclusions"

→ **DCP**: **D**ifferential-**C**omplementarity **P**roblem

O K



Handle large steps, multiple discontinuities

Embed complementarity problem, each step

How to solve complementarity problems?

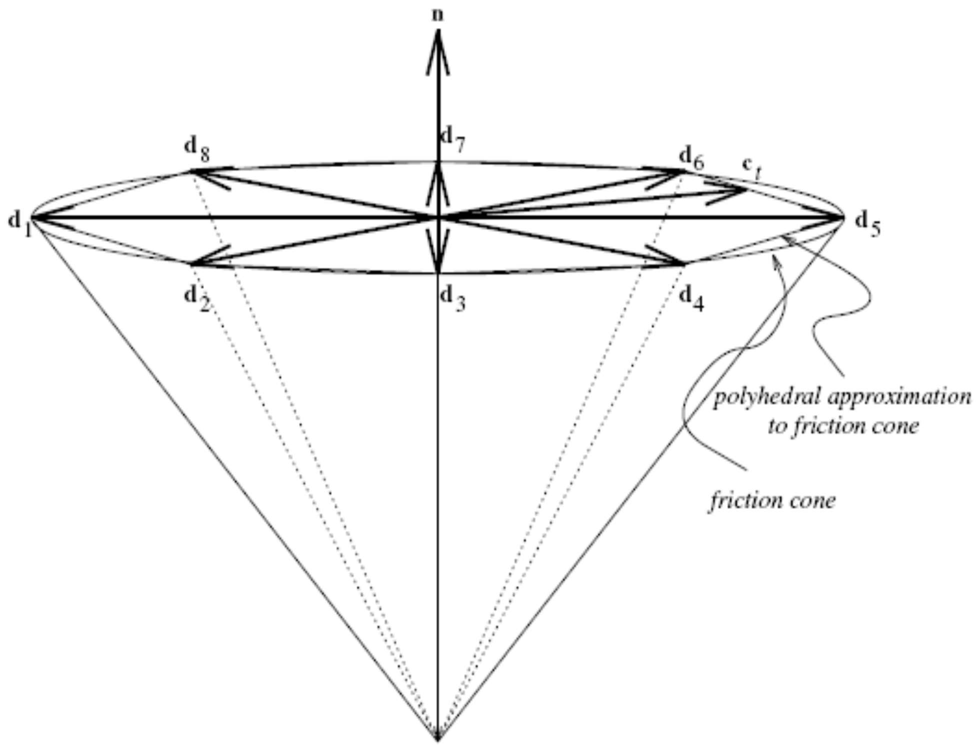
Goal

- Develop a solver for complementarity problems in time stepping methods with full cone formulation
- Must be able to handle thousands or millions of constraints
- Iterative solver, matrix-less, linear-time, linear-space
- Robust: must handle ill-posed and redundant constraints
- Efficient, fast, real-time if possible

Goal

- Why not use off-the-shelf LCP (Linear Complementarity) solvers (PATH)?
- Most LCP solvers (Lemke, Dantzig) are based on exact simplex methods with NP-hard complexity class. Risk of combinatorial explosion!
- *We prefer an approximate O(n) iterativem method*
- LCP require finitely-generated approximations of NL convex sets, see the friction cones:
- *We want no polyhedral approximations, within a matrix-free method for very large scal ecomputation – not available*

Need for a custom iterative **non-linear complementarity solver**



Example (D.Stewart): Polyhedral approximation of friction cones, to feed into typical LCP solvers

Team

- Mihai Anitescu
- Alessandro Tasora
  - University of Parma,
  - Author of ChronoEngine
- Dan Negrut,
  - University of Wisconsin
  - Former ADAMS developer

Nonsmooth contact dynamics

- Differential problem with equilibrium constraints – DPEC.

$$
\begin{aligned}
M\frac{dv}{dt} &= \sum_{j=1,2,\ldots,p}\left(c_n^{(j)}n^{(j)} + \beta_1^{(j)}t_1^{(j)} + \beta_2^{(j)}t_2^{(j)}\right) + f_c(q,v) + k(t,q,v)\\
\frac{dq}{dt} &= v\\
c_n^{(j)} &\geq 0 \quad \perp \quad \Phi^{(j)}(q) \geq 0, \qquad j = 1,2,\ldots,p\\
\left(\beta_1^{(j)},\beta_2^{(j)}\right) &= \operatorname{argmin}_{\mu^{(j)}c_n^{(j)} \geq \sqrt{\left(\beta_1^{(j)}+\beta_2^{(j)}\right)^2}} \left[\left(v^T t_1^{(j)}\right)\beta_1 + \left(v^T t_2^{(j)}\right)\beta_2\right]
\end{aligned}
$$

Friction Model

It is a hybrid system – where is the switching?

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

Pause: Constraint Stabilization

- Compared to original scheme

$$\nabla\Phi(q^{(l)})^T v^{(l+1)} \geq 0 \Longrightarrow \Phi^{(j)}(q^{(l)}) + \gamma h_l \nabla\Phi(q^{(l)})^T v^{(l+1)} \geq 0.$$

$$\nabla\Theta(q^{(l)})^T v^{(l+1)} = 0 \Longrightarrow \Theta^{(j)}(q^{(l)}) + \gamma h_l \nabla\Theta(q^{(l)})^T v^{(l+1)} = 0.$$

- Allows fixed time steps for plastic collisions.
- How do we know it is achieved? Infeasibility is one order better than accuracy (O(h^2))

General: Theory

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

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**

- Answer 2: Simple, but first result of this nature for conic constraints—and HIGHLY EFFICIENT

Conic Complementarity IS NATURAL in Coulomb Models.

- Coulomb model.

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

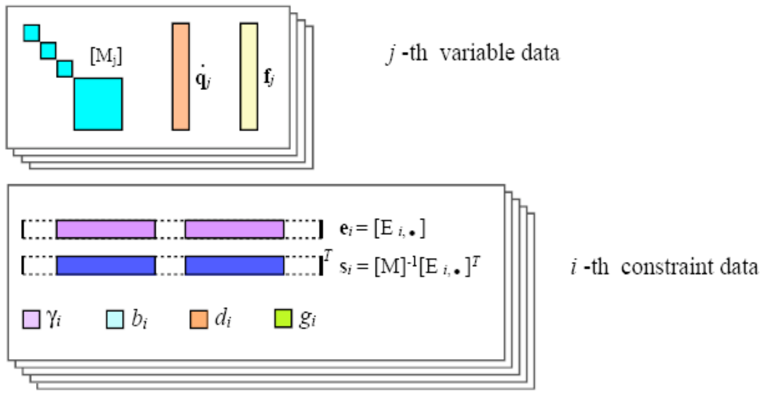
$$K = \left\{(x,y,z) \,\middle|\, \mu^{(j)} z \geq \sqrt{y^2 + x^2}\right\} \quad K^* = \left\{(x,y,z) \,\middle|\, z \geq \mu^{(j)} \sqrt{y^2 + x^2}\right\}$$

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

- Most previous approaches discretize friction cone to use LCP…
- Question 2: Can we still get convergence but not do that?

The algorithm

- Development of an efficient algorithm for fixed point iteration:



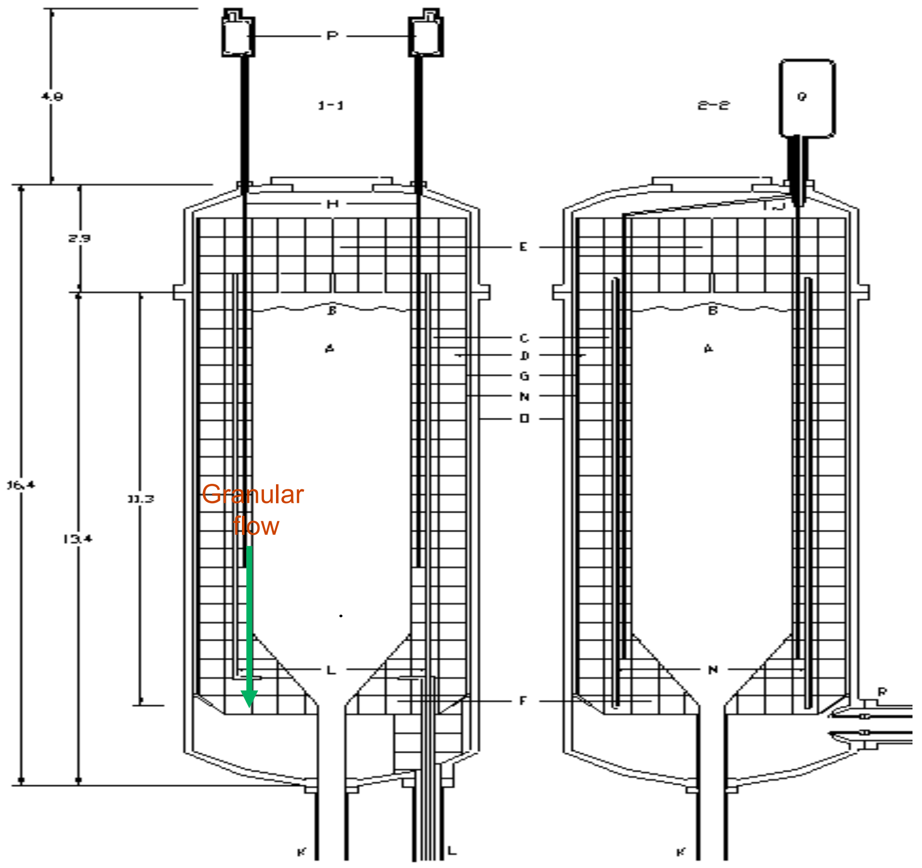
- *avoid temporary data, exploit sparsity. Never compute explicitly the N matrix!*
- *implemented in incremental form. Compute only deltas of multipliers.*
- *O(n) space requirements and supports premature termination*
- *for real-time purposes: O(n) time*

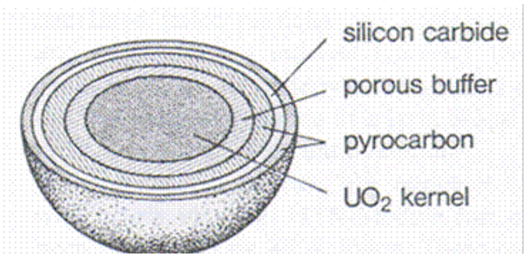
The algorithm is specialized, for minimum memory use!

```
// Pre-compute some data for friction constraints
for $i := 1$ to $n_{\mathcal{A}}$
    $\boldsymbol{s}_a^i = M^{-1} D^i$
    $g_a^i = D^{i,T} \boldsymbol{s}_a^i$
    $\eta_a^i = \frac{3}{\mathrm{Trace}(g_a^i)}$
// Pre-compute some data for bilateral constraints
for $i := 1$ to $n_{\mathcal{B}}$
    $s_b^i = M^{-1} \boldsymbol{\nabla\Psi}^i$
    $g_b^i = \boldsymbol{\nabla\Psi}^{i,T} s_b^i$
    $\eta_b^i = \frac{1}{g_b^i}$

// Initialize impulses
if  warm start with initial guess $\boldsymbol{\gamma}_{\mathcal{E}}^*$
    $\boldsymbol{\gamma}_{\mathcal{E}}^0 = \boldsymbol{\gamma}_{\mathcal{E}}^*$
else
    $\boldsymbol{\gamma}_{\mathcal{E}}^0 = 0$

// Initialize speeds
$\boldsymbol{v} = \sum_{i=1}^{n_{\mathcal{A}}} \boldsymbol{s}_a^i \boldsymbol{\gamma}_a^{i,0} + \sum_{i=1}^{n_{\mathcal{B}}} s_b^i \gamma_b^{i,0} + M^{-1} \tilde{\boldsymbol{k}}$

// Main iteration loop
for $r := 0$ to $r_{max}$
    // Loop on frictional constraints
    for $i := 1$ to $n_{\mathcal{A}}$
        $\boldsymbol{\delta}_a^{i,r} = \left(\boldsymbol{\gamma}_a^{i,r} - \omega\eta_a^i \left(D^{i,T}\boldsymbol{v}^r + \boldsymbol{b}_a^i\right)\right);$
        $\boldsymbol{\gamma}_a^{i,r+1} = \lambda\Pi_{\Upsilon}\left(\boldsymbol{\delta}_a^{i,r}\right) + (1-\lambda)\boldsymbol{\gamma}_a^{i,r}\ ;$
        $\Delta\boldsymbol{\gamma}_a^{i,r+1} = \boldsymbol{\gamma}_a^{i,r+1} - \boldsymbol{\gamma}_a^{i,r}\ ;$
        $\boldsymbol{v} := \boldsymbol{v} + \boldsymbol{s}_a^{i^T} \Delta\boldsymbol{\gamma}_a^{i,r+1}.$
    // Loop on bilateral constraints
    for $i := 1$ to $n_{\mathcal{B}}$
        $\delta_b^{i,r} = \left(\gamma_b^{i,r} - \omega\eta_b^i \left(\nabla\boldsymbol{\Psi}^{i,T}\boldsymbol{v}^r + b_b^i\right)\right);$
        $\gamma_b^{i,r+1} = \lambda\Pi_{\Upsilon}\left(\delta_b^{i,r}\right) + (1-\lambda)\gamma_b^{i,r}\ ;$
        $\Delta\gamma_b^{i,r+1} = \gamma_b^{i,r+1} - \gamma_b^{i,r}\ ;$
        $\boldsymbol{v} := \boldsymbol{v} + s_b^{i^T} \Delta\gamma_b^{i,r+1}.$

return $\boldsymbol{\gamma}_{\mathcal{E}}$, $\boldsymbol{v}$
```

Simulating the PBR nuclear reactor

- The PBR 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 360’000 graphite spheres in a pebble bed

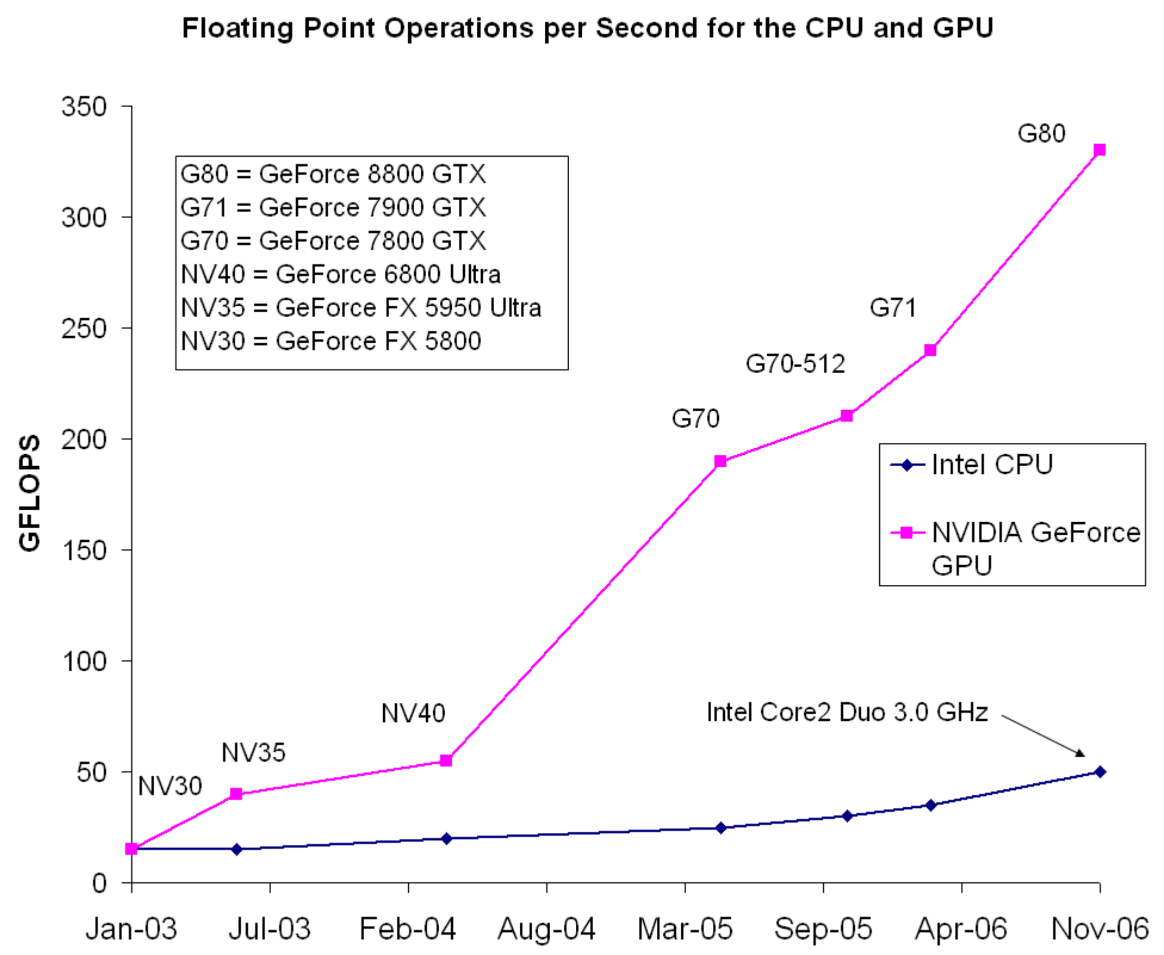




Options and challenges for methods with no smoothing

- Piecewise DAE (Haug, 86)
  - Plus : Uses well understood DAE technology
  - Minus: The density of switches, switching consistency, and Painleve are problems.
- Acceleration-force time-stepping (Glocker & Pfeiffer, 1992, Pang & Trinkle, 1995)
  - Plus: No consistency problem.
  - Minus: Density of switches and Painleve.
- Velocity-impulse time-stepping. (Moreau, 196*, 198*,199*, Stewart and Trinkle, 1996, Anitescu & Potra, 1997)
  - 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

New large scale computational opportunity Graphical Processing Unit



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!!!

Brick Wall Example…

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

THE UNIVERSITY of WISCONSIN MADISON

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