GPU TECHNOLOGY CONFERENCE:

S5400: Chrono::SPIKE – A Nonsmooth Contact Dynamics Framework on the GPU

Daniel Melanz, Luning Fang, Ang Li, Hammad Mazhar, Radu Serban, Dan Negrut

Simulation-Based Engineering Laboratory
University of Wisconsin - Madison

SBEL Wisconsin Applied Computing Center

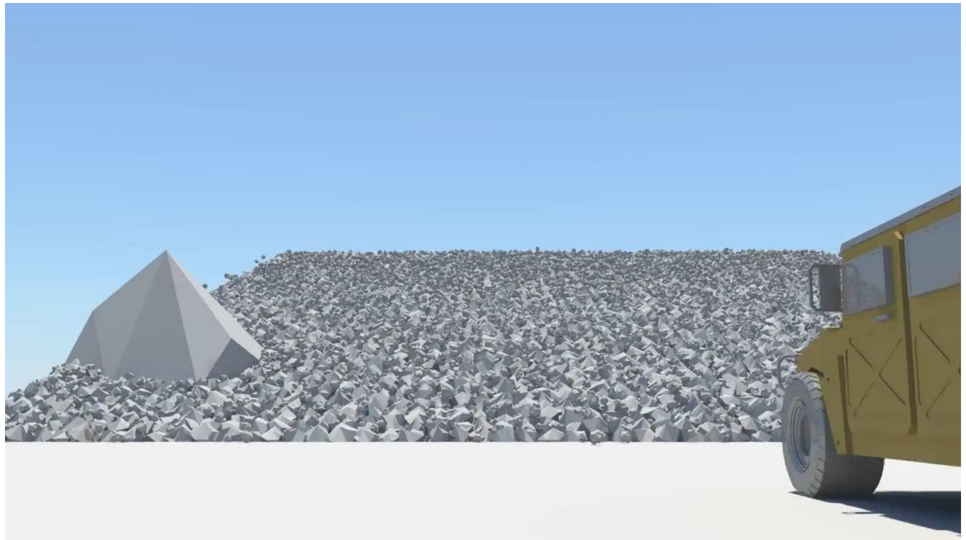
Overview

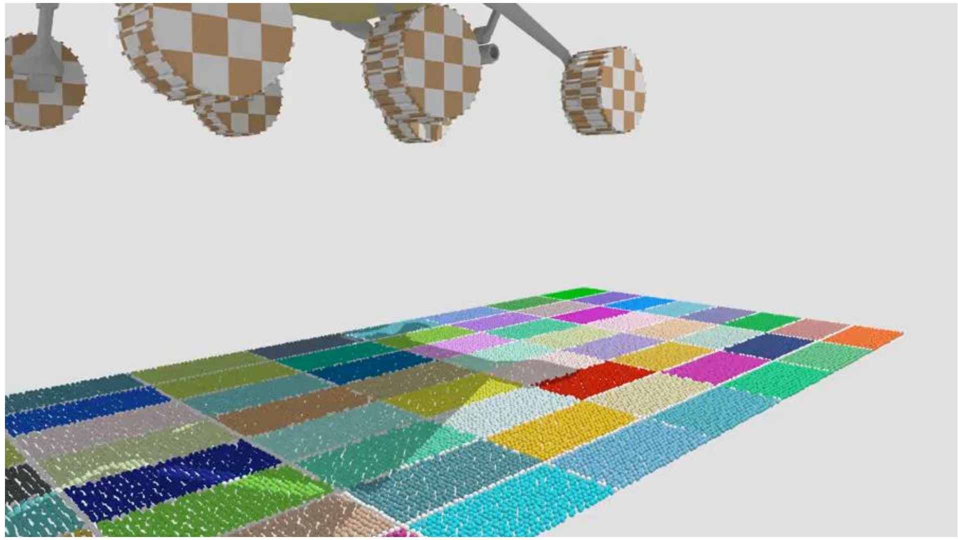
1) Nonsmooth Contact Dynamics
2) Quadratic Optimization w/ Conic Constraints
3) Preconditioning with SPIKE
4) Numerical Results
5) Conclusions & Future Work

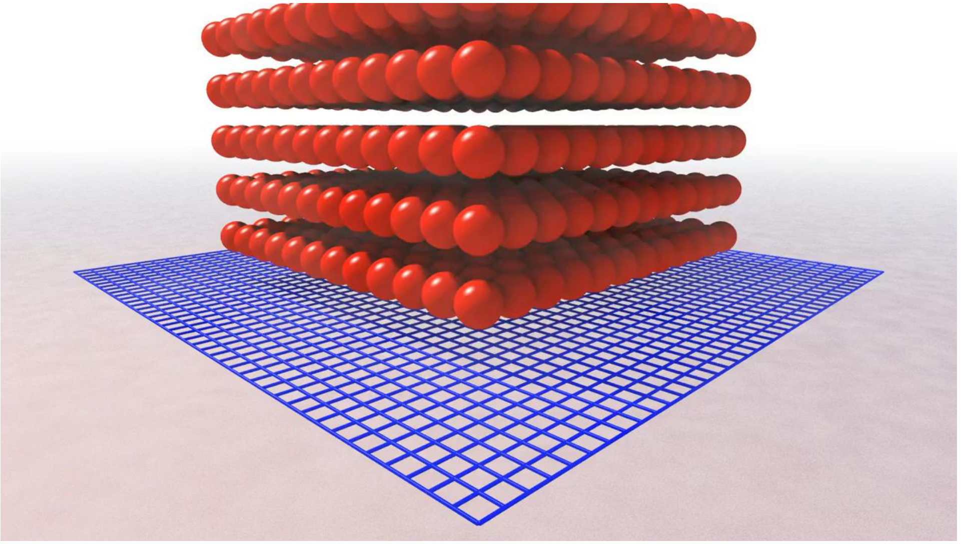
Nonsmooth Contact Dynamics

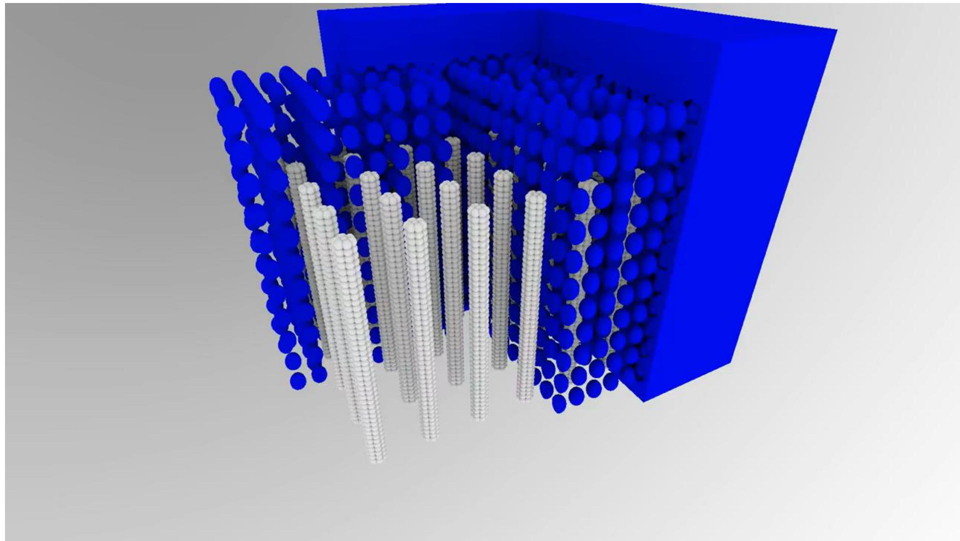
Wisconsin Applied Computing Center

Nonsmooth Dynamics









Wisconsin Applied Computing Center

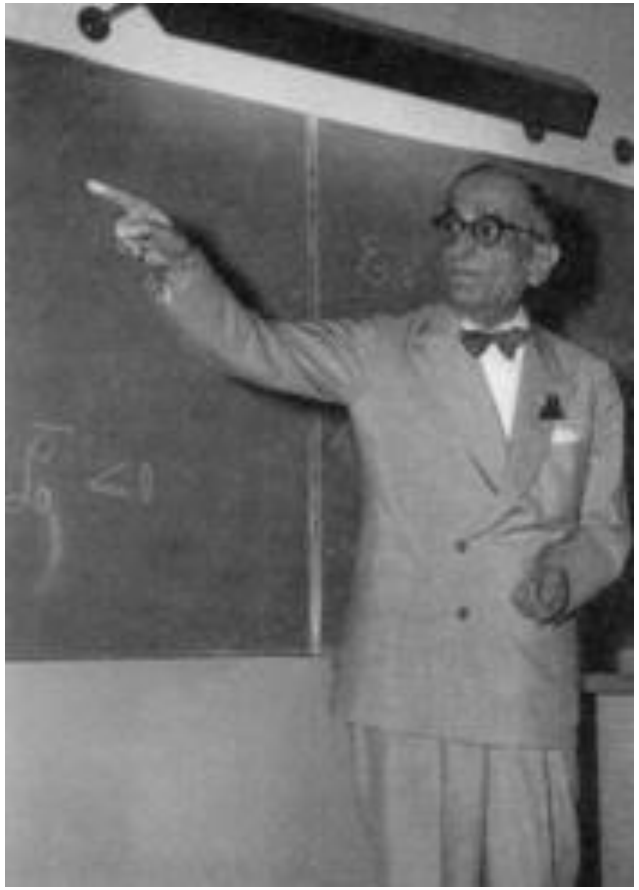
Nonsmooth Dynamics: Frictionless Case

The **Signorini Conditions**:

$0 \leq \mathbf{v}_{rel}$ Every relative velocity should be zero or separating

$0 \leq \lambda$ Every contact impulse should be non-attractive

No impulse at separating contacts: $(\mathbf{v}_{rel})_i = 0 \ \text{ or } \ \lambda_i = 0$



Antonio Signorini

Tonge, 2012

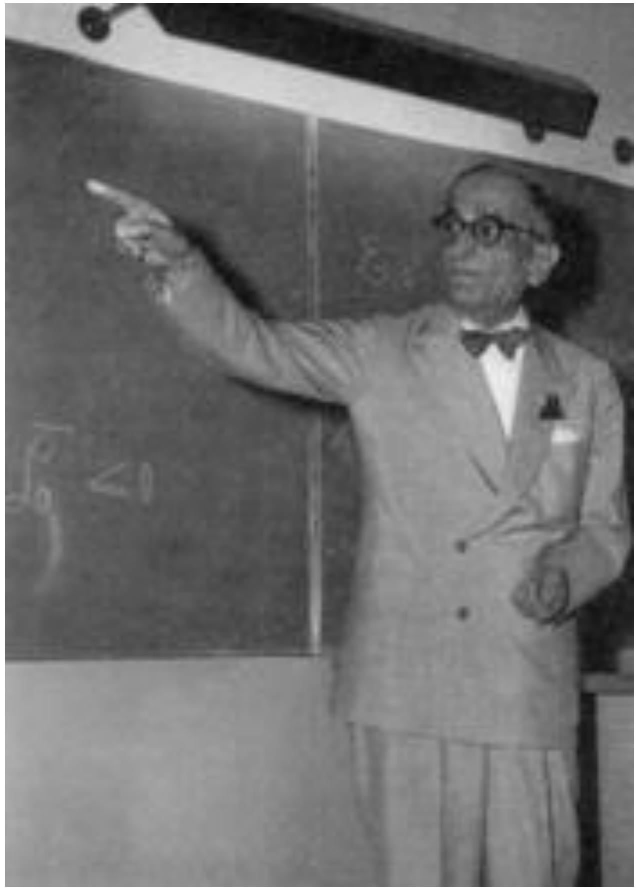
SBEL
Wisconsin Applied Computing Center

Nonsmooth Dynamics: Frictionless Case

The **Signorini Conditions**:

$$0 \leq \mathbf{v}_{\mathrm{rel}} \perp 0 \leq \lambda$$

This is a compact way to write the three conditions in one line of math



Antonio Signorini

Tonge, 2012

Nonsmooth Dynamics: Frictionless Case

The final model can be expressed by these equations:

$$\mathbf{M}\ddot{\mathbf{x}} = \mathbf{J}^T\lambda + \mathbf{f}_e$$

$$\dot{\mathbf{x}} = \mathbf{v}$$

$$\mathbf{0} \leq \lambda \perp \mathbf{0} \leq \mathbf{J}\mathbf{v}$$

Tonge, 2012

Nonsmooth Dynamics: Friction Case

Generalized Positions

Generalized Mass Matrix

Velocity Transformation Matrix

Generalized Velocities

Kinematic Differential Equations → $\dot{\mathbf{q}} = \mathbf{T}(\mathbf{q})\mathbf{v}$

Reaction Force

Frictional Contact Force

Force Balance Equations → $\mathbf{M}(\mathbf{q})\dot{\mathbf{v}} = \mathbf{f}(t,\mathbf{q},\mathbf{v}) - \mathbf{g}_{\mathbf{q}}^{\mathrm{T}}(\mathbf{q},t)\hat{\lambda} + \sum_{i=1}^{N_c} (\hat{\gamma}_n^i \mathbf{D}_n^i + \hat{\gamma}_u^i \mathbf{D}_u^i + \hat{\gamma}_w^i \mathbf{D}_w^i)$

Applied Force

Holonomic Kinematic Constraints → $\mathbf{g}(\mathbf{q},t) = \mathbf{0}$

Normal Force, for Contact "i"

Contact Complementarity Conditions → $0 \le \Phi^i(\mathbf{q},t) \perp \hat{\gamma}_n^i \ge 0 \qquad i = 1,2,\ldots,N_c$

Total Number of Contacts

Coulomb Friction Model → $(\hat{\gamma}_u^i, \hat{\gamma}_w^i) = \underset{\mu^i \hat{\gamma}_n^i \ge \sqrt{(\bar{\gamma}_u^i)^2 + (\bar{\gamma}_w^i)^2}}{\arg\min} (\bar{\gamma}_u^i \mathbf{v}^T \mathbf{D}_u^i + \bar{\gamma}_w^i \mathbf{v}^T \mathbf{D}_w^i)$

Friction Dissipation Energy

Gap Function, for Contact "i"

Friction Components, for Contact "i"

Stewart and Trinkle, 1996

Nonsmooth Dynamics: Friction Case

Time Step Index; Generalized Positions; Generalized Speeds; Time Step Size; Velocity Transformation Matrix

Kinematic Differential Equations

$$\mathbf{q}^{(l+1)} = \mathbf{q}^{(l)} + h\mathbf{T}(\mathbf{q}^{(l)})\mathbf{v}^{(l+1)}$$

Reaction Impulse; Frictional Contact Reaction Impulses; Applied Force

Force Balance Equations

$$\mathbf{M}\left(\mathbf{v}^{(l+1)} - \mathbf{v}^{(l)}\right) = h\mathbf{f}(t^{(l)}, \mathbf{q}^{(l)}, \mathbf{v}^{(l)}) - \mathbf{g}_{\mathbf{q}}^{T}(\mathbf{q}^{(l)}, t)\boldsymbol{\lambda} + \sum_{i=1}^{N_c}(\gamma_n^i \mathbf{D}_n^i + \gamma_u^i \mathbf{D}_u^i + \gamma_w^i \mathbf{D}_w^i)$$

Stabilization Terms

Holonomic Kinematic Constraints

$$\frac{1}{h}\mathbf{g}(\mathbf{q}^{(l)}, t) + \mathbf{g}_{\mathbf{q}}{}^{T}\mathbf{v}^{(l+1)} + \mathbf{g}_t = \mathbf{0}$$

Contact Complementarity Conditions

$$0 \le \frac{1}{h}\Phi^i(\mathbf{q}^{(l)}, t) + \mathbf{D}_n^{T,i}\mathbf{v}^{(l+1)} \perp \gamma_n^i \ge 0 \qquad i = 1, 2, \ldots, N_c$$

Total Number of Contacts

Coulomb Friction Model

$$(\gamma_u^i, \gamma_w^i) = \underset{\mu^i \gamma_n^i \ge \sqrt{(\bar{\gamma}_u^i)^2 + (\bar{\gamma}_w^i)^2}}{\arg\min} (\bar{\gamma}_u^i \mathbf{v}^T \mathbf{D}_u^i + \bar{\gamma}_w^i \mathbf{v}^T \mathbf{D}_w^i)$$

Anitescu and Hart, 2004

Nonsmooth Dynamics: The Cone Complementarity Problem (CCP)

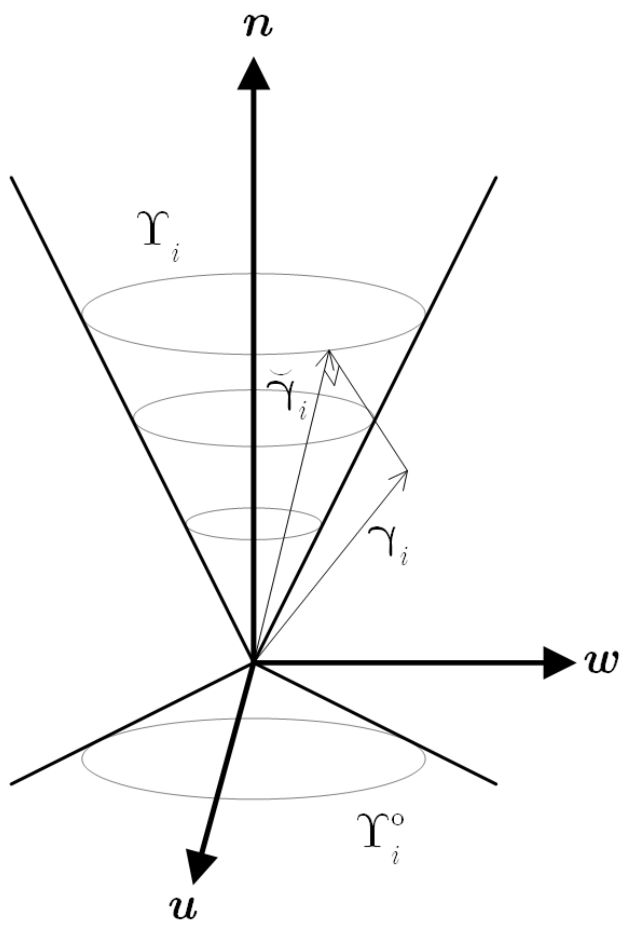
$$\text{Find } \boldsymbol{\gamma}_i^{(l+1)}, \text{ for } i = 1, \ldots, N_c$$

$$\text{such that } \Upsilon_i \ni \boldsymbol{\gamma}_i^{(l+1)} \perp -\left(\boldsymbol{N}\boldsymbol{\gamma}^{(l+1)} + \boldsymbol{r}\right)_i \in \Upsilon_i^\circ$$

where

$$\Upsilon_i = \{[x, y, z]^T \in \mathbb{R}^3 | \sqrt{y^2 + z^2} \le \mu_i x\}$$

$$\Upsilon_i^\circ = \{[x, y, z]^T \in \mathbb{R}^3 | x \le -\mu_i \sqrt{y^2 + z^2}\}$$



Nonsmooth Dynamics: The Quadratic Programming Angle…

- The CCP captures the first-order optimality condition for a quadratic optimization problem with conic constraints:

$$\min \boldsymbol{q}(\boldsymbol{\gamma}) = \frac{1}{2}\boldsymbol{\gamma}^T \boldsymbol{N}\boldsymbol{\gamma} + \boldsymbol{r}^T\boldsymbol{\gamma}$$
$$\text{subject to } \boldsymbol{\gamma}_i \in \Upsilon_i \text{ for } i = 1, 2, \ldots, N_c$$

- Notation used:

$$\begin{aligned} \boldsymbol{N} &= \boldsymbol{D}^T \boldsymbol{M}^{-1} \boldsymbol{D} \\ \boldsymbol{r} &= \boldsymbol{b} + \boldsymbol{D}^T \boldsymbol{M}^{-1} \boldsymbol{k} \end{aligned}$$

$$\boldsymbol{\gamma} = \left[\boldsymbol{\gamma}_1^T, \boldsymbol{\gamma}_2^T, \ldots, \boldsymbol{\gamma}_{N_c}^T\right]^T \in \mathbb{R}^{3N_c}$$

Quadratic Optimization w/ Conic Constraints (CCQO's)

CCQO's: First Order Methods

ALGORITHM JACOBI($\boldsymbol{N}, \boldsymbol{r}, \tau, N_{max}, \boldsymbol{\gamma}_0$)

(1) **for** $k := 0$ **to** N_{max}

(2) $\hat{\boldsymbol{\gamma}}_{(k+1)} = \Pi_{\mathcal{K}}\left(\boldsymbol{\gamma}_{(k)} - \omega \boldsymbol{B}\left(\boldsymbol{N}\boldsymbol{\gamma}_{(k)} + \boldsymbol{r}\right)\right)$

(3) $\boldsymbol{\gamma}_{(k+1)} = \lambda \hat{\boldsymbol{\gamma}}_{(k+1)} + (1-\lambda)\,\boldsymbol{\gamma}_{(k)}$

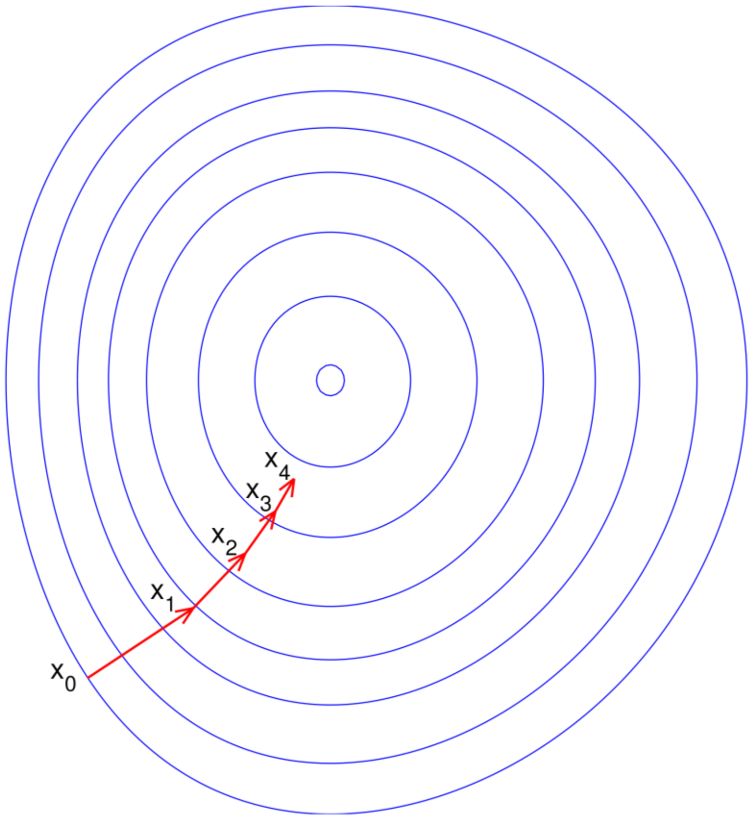
(4) $r = r\left(\boldsymbol{\gamma}_{(k+1)}\right)$

(5) **if** $r < \tau$

(6) **break**

(7) **endfor**

(8) **return** Value at time step $t^{(l+1)}$, $\boldsymbol{\gamma}^{(l+1)} := \boldsymbol{\gamma}_{(k+1)}$



CCQO's: Second Order Methods

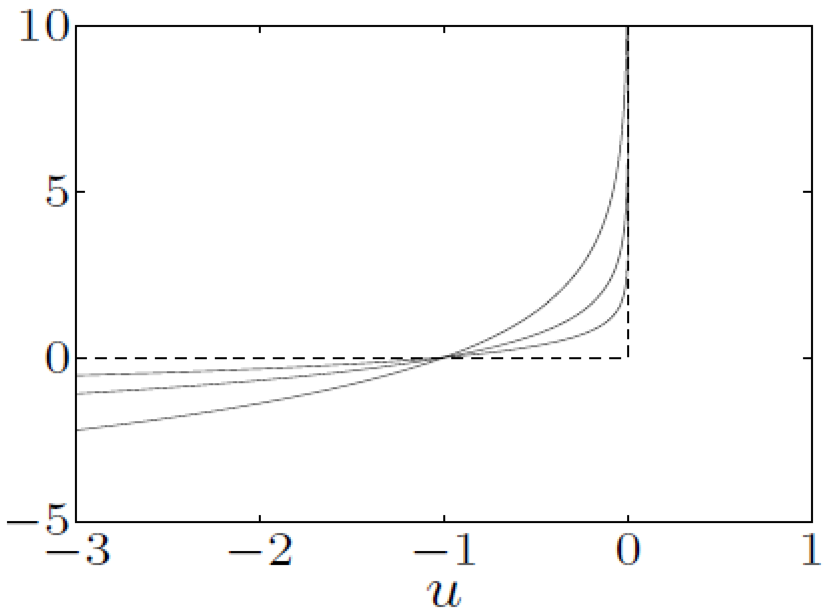
- Original problem:

$$\begin{array}{ll}\text{minimize} & f_0(x) \\ \text{subject to} & f_i(x) \le 0, \quad i = 1, \dots, m \\ & Ax = b\end{array}$$

- Reformulation via an indicator function:

$$\begin{array}{ll}\text{minimize} & f_0(x) + \sum_{i=1}^{m} I_{-}(f_i(x)) \\ \text{subject to} & Ax = b\end{array}$$

where $I_{-}(u) = 0$ if $u \le 0$, $I_{-}(u) = \infty$ otherwise



- Approximation via logarithmic barrier:

$$\begin{array}{ll}\text{minimize} & f_0(x) - (1/t) \sum_{i=1}^{m} \log(-f_i(x)) \\ \text{subject to} & Ax = b\end{array}$$

Interior Point

ALGORITHM PD-IP($f_0, f_1, \ldots, f_m, \mu \geq 1, \epsilon$)

(1) **while** $||\boldsymbol{r}_t(\boldsymbol{x}, \boldsymbol{\lambda})||_2 > \epsilon$

(2) Compute $t = \frac{\mu m}{\hat{\eta}}$

(3) Compute search direction $[\Delta \boldsymbol{x}^T \; \Delta \boldsymbol{\lambda}^T]^T$

(4) Compute step length $s > 0$ via line search

(5) Update: $\boldsymbol{x} = \boldsymbol{x} + s\Delta\boldsymbol{x}, \boldsymbol{\lambda} = \boldsymbol{\lambda} + s\Delta\boldsymbol{\lambda}$

(6) **endwhile**

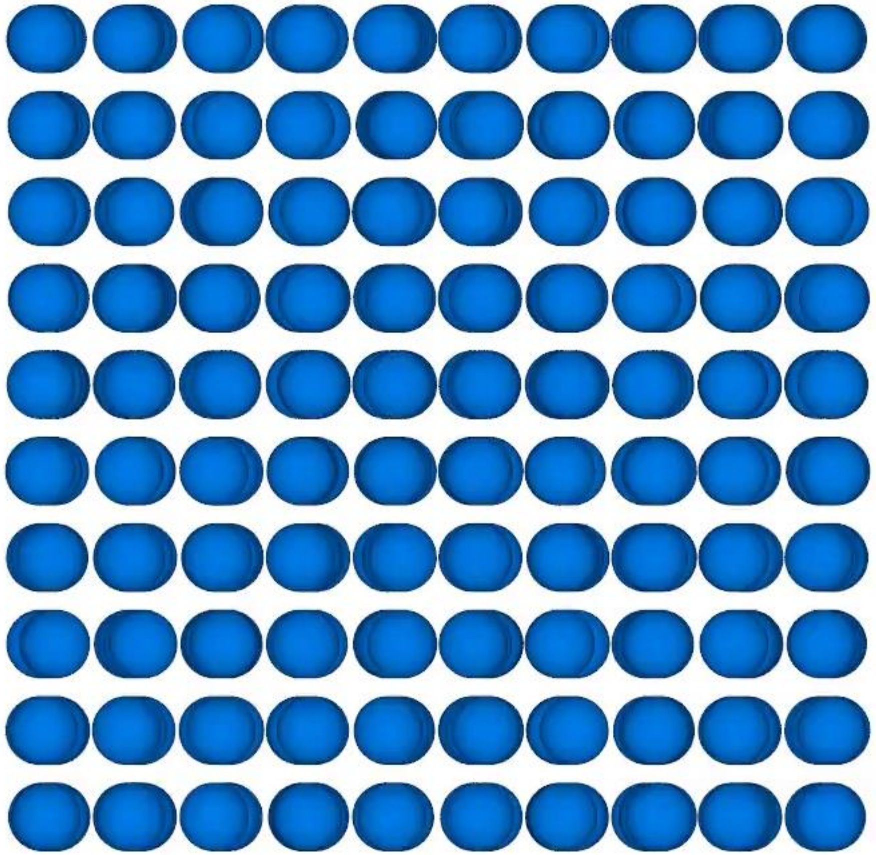
(7) **return** Solution $\boldsymbol{x}^\star = \boldsymbol{x}, \boldsymbol{\lambda}^\star = \boldsymbol{\lambda}$

$$\begin{bmatrix} \nabla^2 f_0(\boldsymbol{x}) + \sum_{i=1}^{m} \boldsymbol{\lambda}_i \nabla^2 \boldsymbol{f}_i(\boldsymbol{x}) & \nabla \boldsymbol{f}(\boldsymbol{x})^T \\ -diag(\boldsymbol{\lambda}) \nabla \boldsymbol{f}(\boldsymbol{x}) & -diag(\boldsymbol{f}(\boldsymbol{x})) \end{bmatrix} \begin{bmatrix} \Delta \boldsymbol{x} \\ \Delta \boldsymbol{\lambda} \end{bmatrix} = -\begin{bmatrix} \nabla f_0(\boldsymbol{x}) + \nabla \boldsymbol{f}(\boldsymbol{x})^T \boldsymbol{\lambda} \\ -diag(\boldsymbol{\lambda}) \boldsymbol{f}(\boldsymbol{x}) - \frac{1}{t}\mathbf{1} \end{bmatrix}$$

Numerical Results

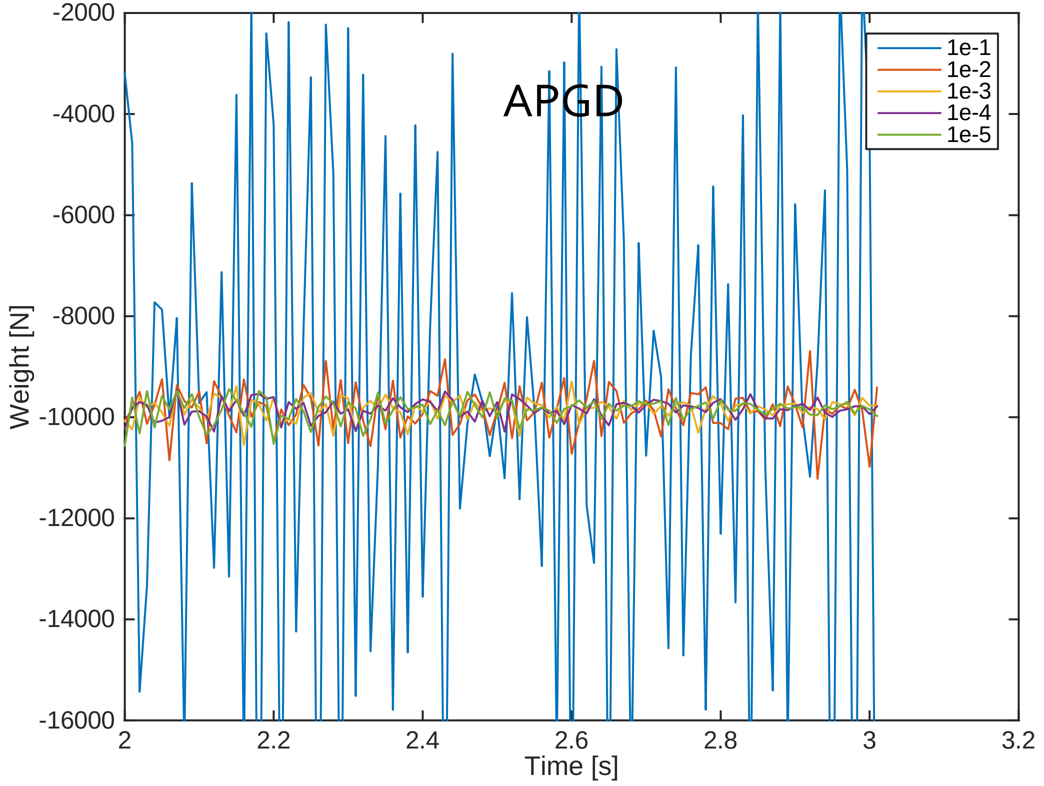
Results: Physical Model

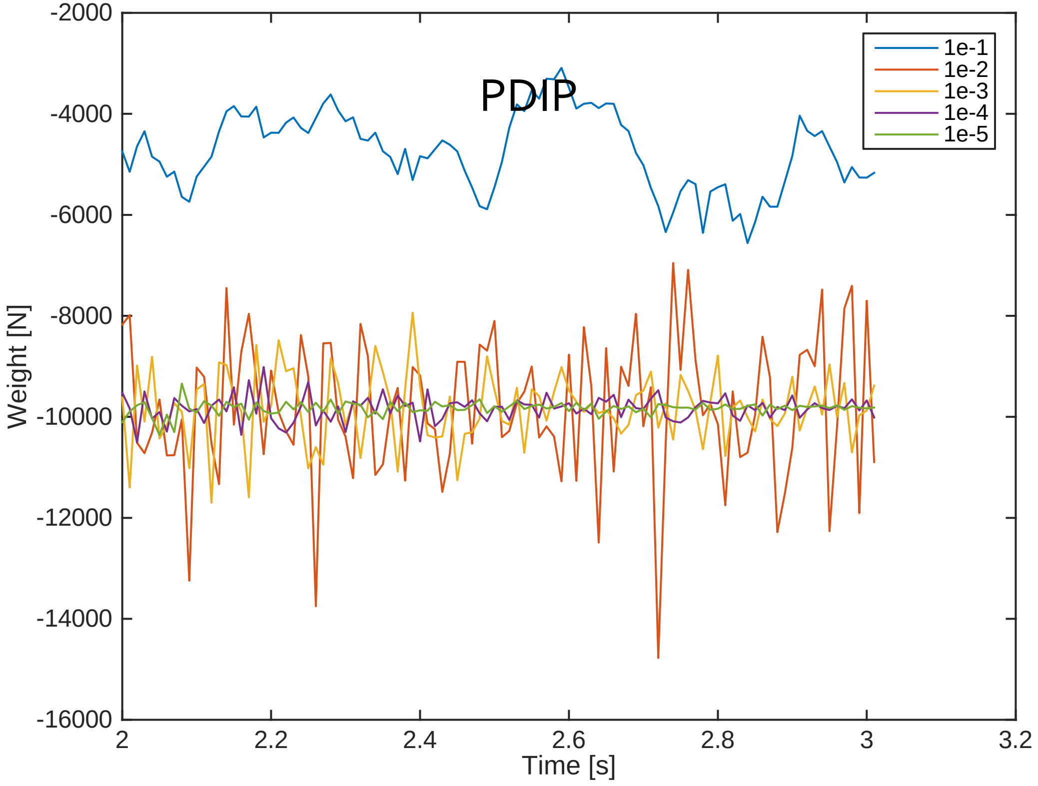
- Several numerical experiments were performed using a model of spheres falling into a bucket



Results: Comparison of Solver Results

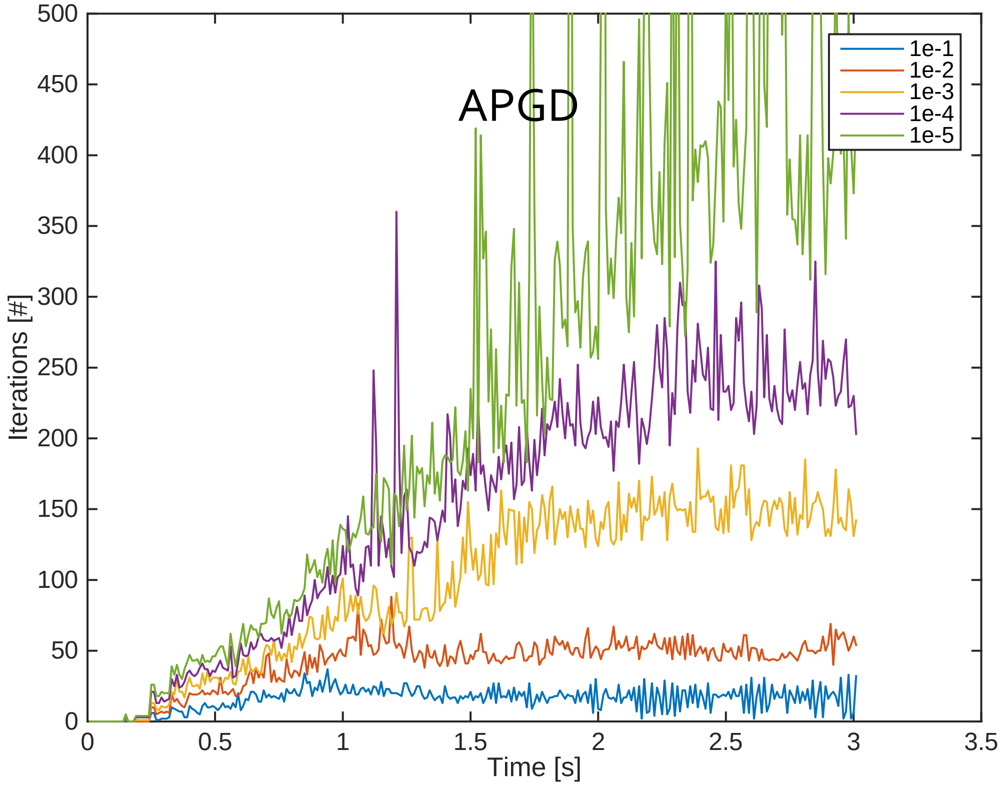
- Simulations of the filling simulation were performed for 3 seconds with a step size, $h=10^{-3}$ seconds using the APGD and PDIP solvers

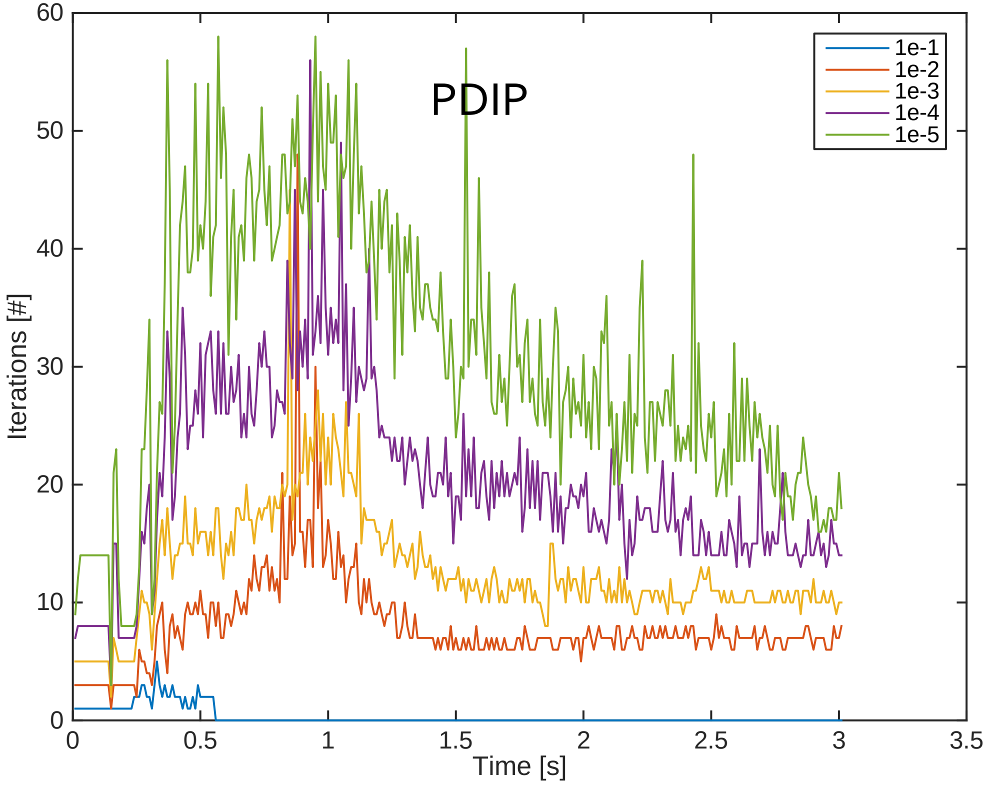




Results: Comparison of Solver Iterations

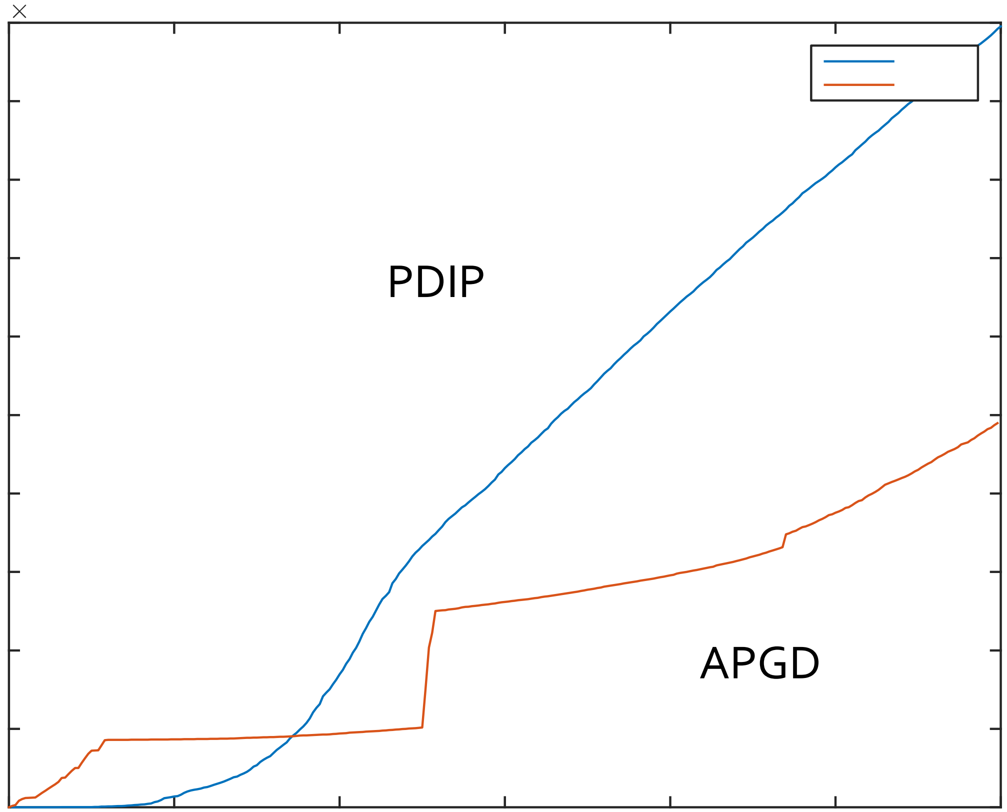
- Simulations of the filling simulation were performed for 3 seconds with a step size, h=10^{-3} seconds using the APGD and PDIP solvers





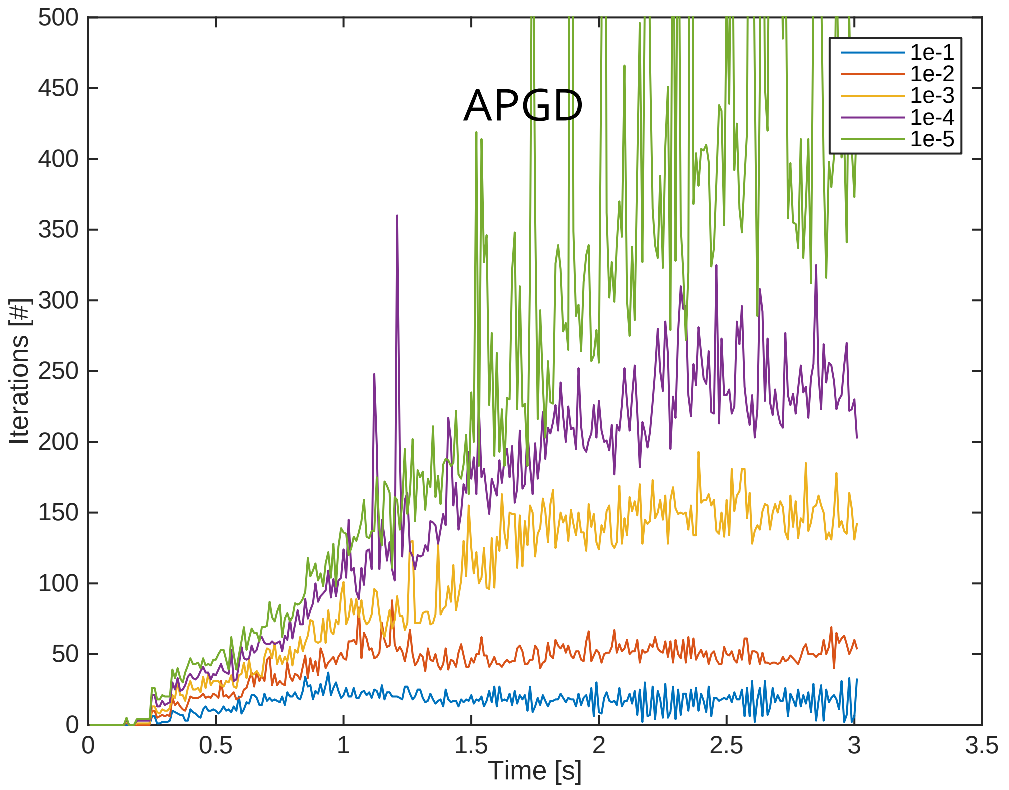
Results: Comparison of Solver Execution Time

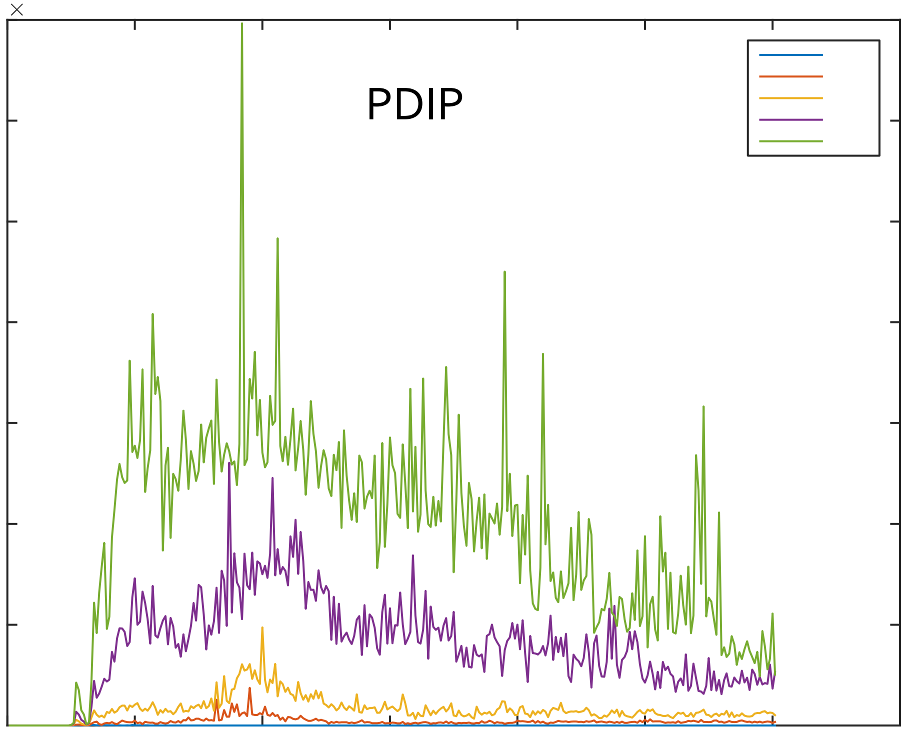
- Simulations of the filling simulation were performed for 3 seconds with a step size, $h=10^{-3}$ seconds using the APGD and PDIP solvers



Results: Comparison of Solvers

- Simulations of the filling simulation were performed for 3 seconds with a step size, h=10^{-3} seconds using the APGD and PDIP solvers





Wisconsin Applied Computing Center

Preconditioning with SPIKE

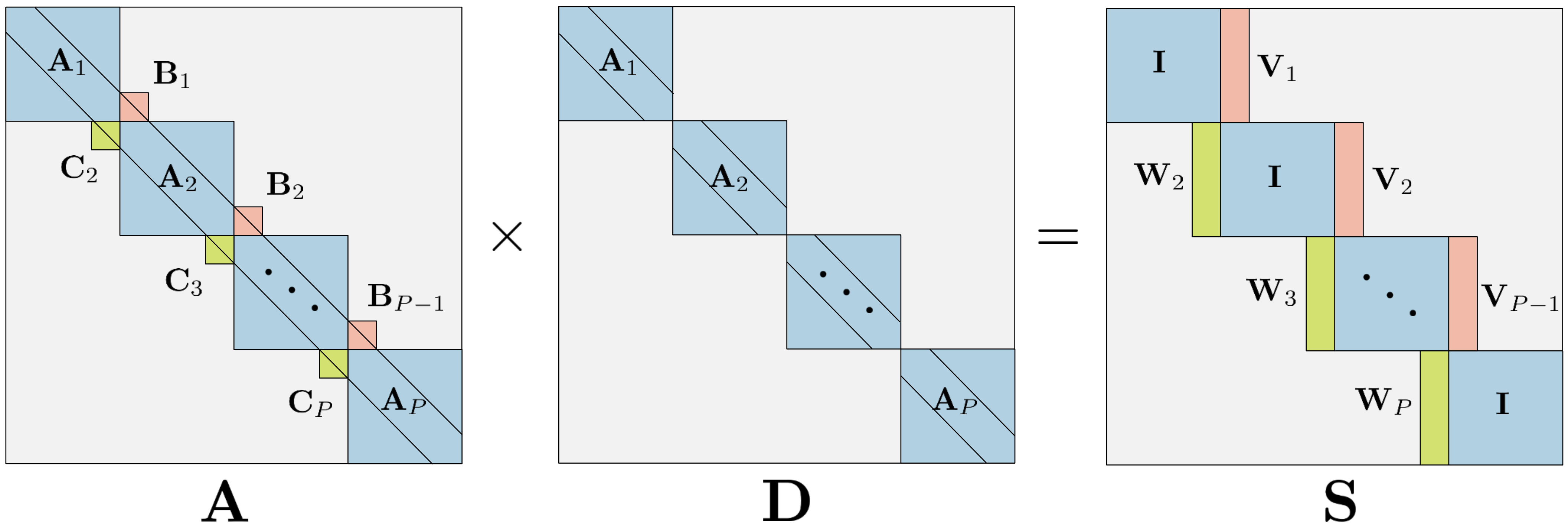
The SPIKE algorithm

- SPIKE: a divide-and-conquer approach to solving **banded** dense systems.
- Proposed by A. H. Sameh and D. J. Kuck in 1978.
  (see also E. Polizzi and A. H. Sameh, *Parallel Computing 32(2)*, 2006)
- Basic idea:
  - Partition the matrix $\mathbf{A}$.
  - Factorize $\mathbf{A}$ to isolate independent blocks.
  - Solve a reduced system to account for coupling information.
  - Recover solution of original system.
- SPIKE comes in two main flavors:
  - **Full-SPIKE**: recursively solve an exact reduced system (direct solver for banded matrices).
  - **Truncated-SPIKE**: solve an approximate reduced system in one step (needs iterative refinement).

SPIKE: algorithmic details
Partitioning and Factorization

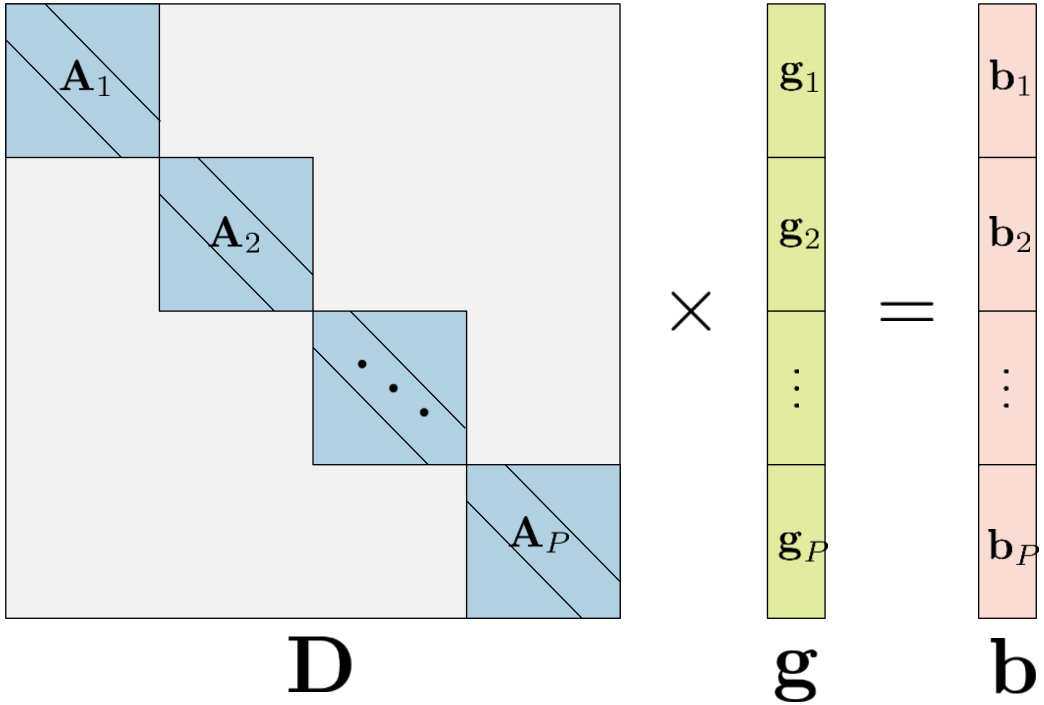
- Partition and factorize $\mathbf{A}$ into *block diagonal* matrix $\mathbf{D}$ and *spike* matrix $\mathbf{S}$.

$$\mathbf{Ax} = \mathbf{b} \quad \Leftrightarrow \quad \begin{cases} \mathbf{Dg} = \mathbf{b} \\ \mathbf{Sx} = \mathbf{g} \end{cases}$$



SPIKE: algorithmic details
Solving $\mathbf{Dg}=\mathbf{b}$

- Reduced to solving P independent (banded dense) linear systems.
- Map these systems to P blocks on GPU.
- Apply classical LU (or UL) methods to each sub-system.



SPIKE factorization in plain math

- The right ($\mathbf{V}_i$) and left ($\mathbf{W}_i$) spike blocks can be obtained through the solution of P independent multiple-RHS banded linear systems.

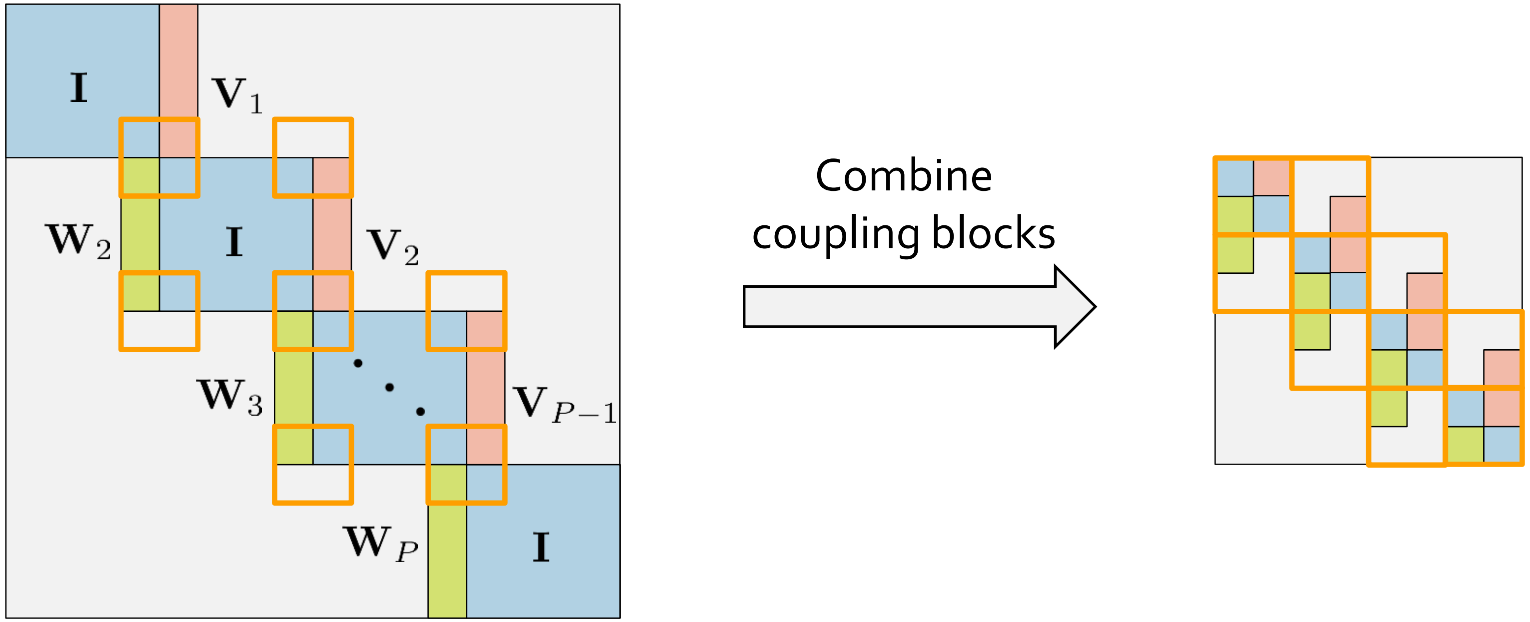
$$\mathbf{A}_1 \mathbf{V}_1 = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{B}_1 \end{bmatrix}$$

$$\mathbf{A}_i \left[\mathbf{W}_i \mid \mathbf{V}_i \right] = \begin{bmatrix} \mathbf{C}_i & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_i \end{bmatrix}, \quad i = 2, \ldots, P-1$$

$$\mathbf{A}_P \mathbf{W}_P = \begin{bmatrix} \mathbf{C}_P \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}.$$

SPIKE: algorithmic details
Solving $\mathbf{Sx=g}$ (full SPIKE)

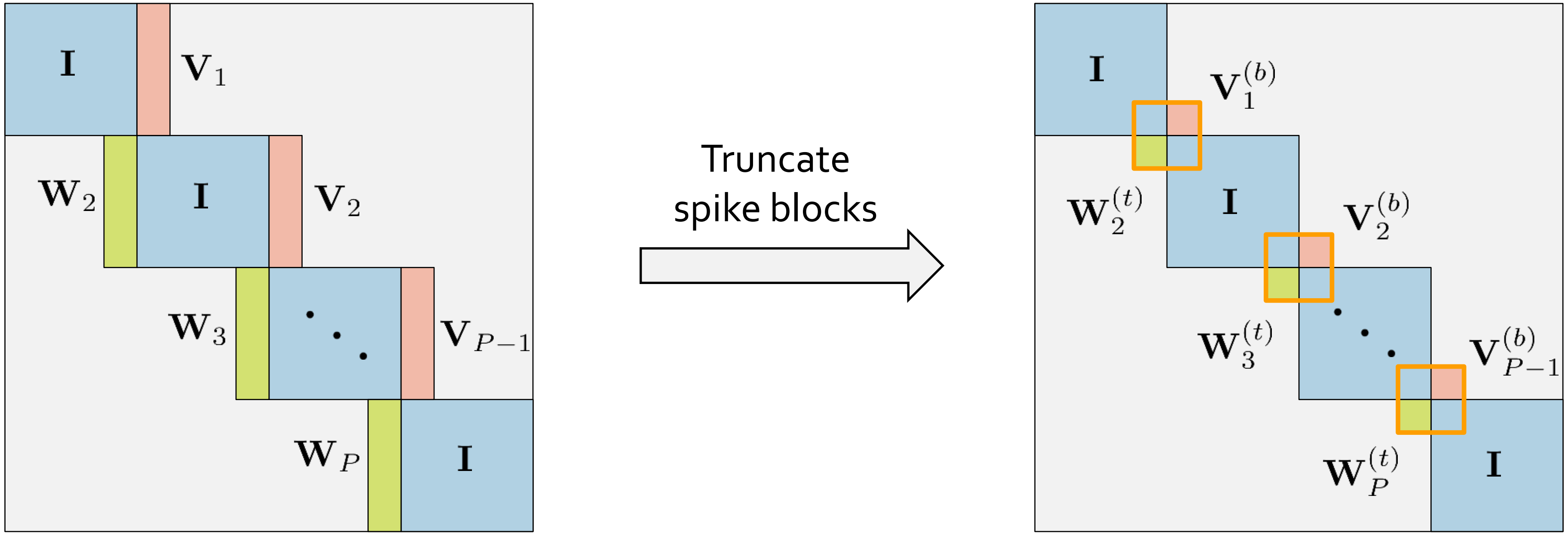
- Combine all coupling blocks into a *reduced* matrix
- (Recursively) solve the reduced system
- Recover solution from reduced solution



SPIKE: algorithmic details

Solving $\mathbf{Sx}=\mathbf{g}$ (truncated SPIKE)

- Justified for **diagonally dominant** systems only.
- All spike blocks $\mathbf{W}$ and $\mathbf{V}$ are approximated by their top and bottom parts, respectively.
- Results in a decoupling of the reduced matrix into (P-1) small independent systems ($2K$ x $2K$).



Truncated SPIKE as a preconditioner

- Fundamental idea:
  - Reorder a sparse matrix to obtain a banded matrix with as "heavy" a diagonal as possible.
  - Drop small entries far from the main diagonal in an attempt to produce an even narrower band.
  - Use truncated SPIKE on resulting banded matrix.
- Sparse matrix reordering
  - Reordering is critical
    - Non-zeroes can spread while we prefer them to gather around diagonals.
    - Both truncated SPIKE and BiCGStab(2) prefer diagonal elements with large absolute values.
  - Reordering strategies
    - Use row permutations to maximize product of absolute diagonal values: $\mathbf{A} \to \mathbf{QA}$
    - Apply symmetric RCM for bandwidth reduction: $\mathbf{QA} + \mathbf{A}^{\mathrm{T}}\mathbf{Q}^{\mathrm{T}} \to \mathbf{P}\,(\,\mathbf{QA} + \mathbf{A}^{\mathrm{T}}\mathbf{Q}^{\mathrm{T}}\,)\,\mathbf{P}^{\mathrm{T}}$

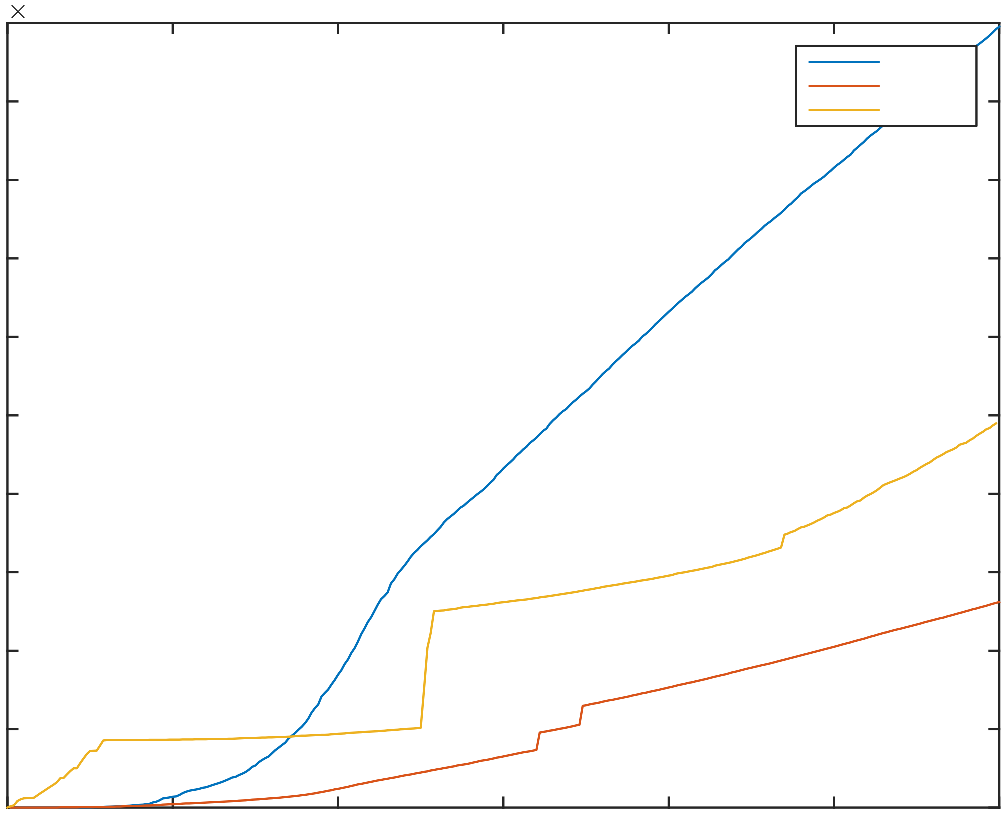
SBEL
Wisconsin Applied Computing Center

Numerical Results

Wisconsin Applied Computing Center

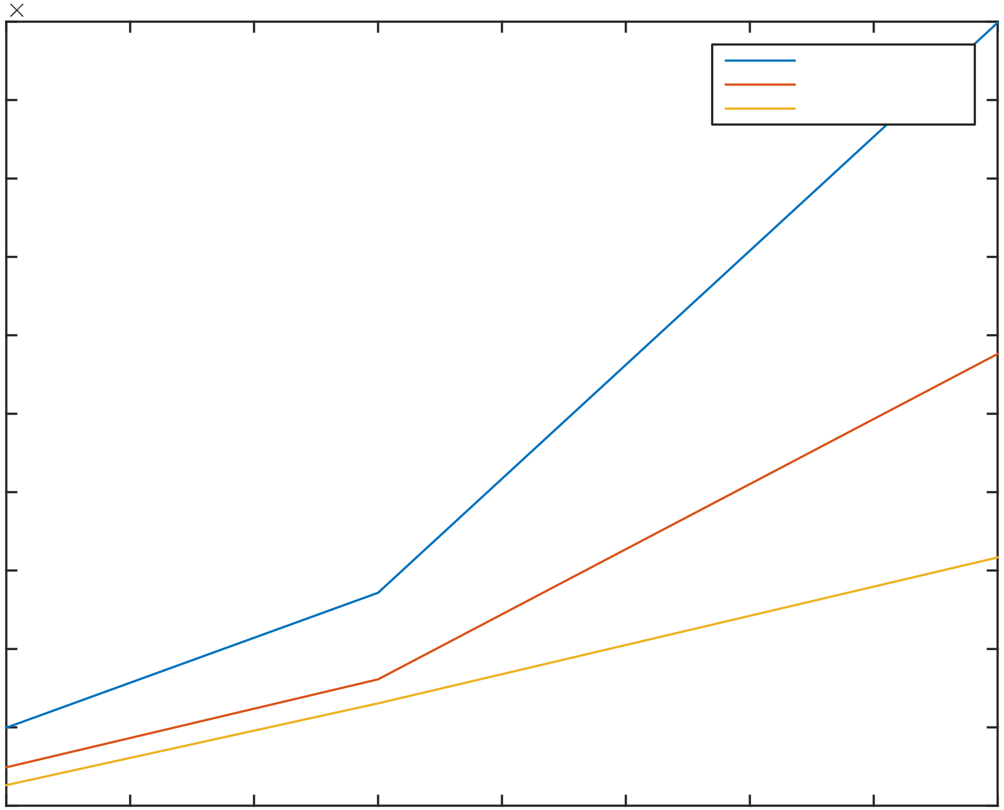
Results: Preconditioned PDIP (P-PDIP)

- Adding preconditioning to the search direction computation drastically improves computation time



Results: Effect of Problem Size

- A series of simulations on filling models of increasing size were performed to estimate how the solver performance scales with problem dimension



Conclusions & Future Work

Conclusions

- Interior point methods require much less iterations than gradient descent methods, but each iteration is much more computationally expensive
- Preconditioning is responsible for an four-fold reduction in run times when simulating nonsmooth contact problems
- Although used with the nonsmooth dynamics, this speed-up is independent of the specific formalism adopted for the formulation of the equations of motion

Future Work

- Investigate improvements to the interior point algorithm
- Investigate SPIKE update strategies and preconditioner re-use
- Investigate the effectiveness of spectral reordering methods
- Understand and gauge the software implementation effort and simulation efficiency trade-offs related to moving from the GPU to parallel multi-core CPU architectures

Thank you.

- Source available for download under BSD-3
  http://spikegpu.sbel.org/
- For all of our animations, please visit
  https://vimeo.com/uwsbel
- For more information about the Simulation-Based Engineering Laboratory, please visit
  http://sbel.wisc.edu/

Wisconsin Applied Computing Center

Thank You.

melanz@wisc.edu

Simulation Based Engineering Lab

Wisconsin Applied Computing Center