A Multigrid Fluid Pressure Solver Handling Separating Solid Boundary Conditions

Nuttapong Chentanez

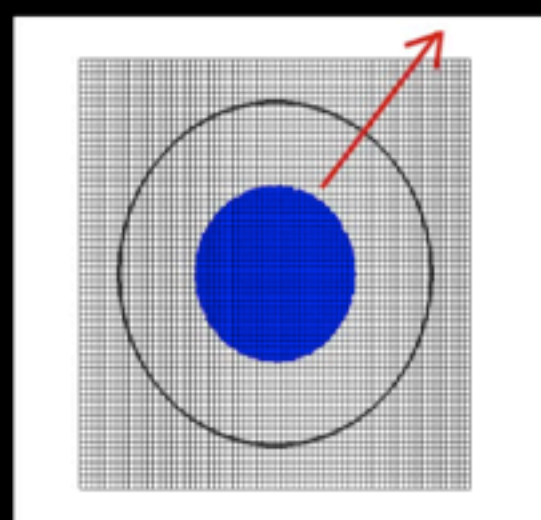
Matthias Müller

Main Contributions

- Multigrid method for solving the weighted Poisson equations
  - From the variational framework for fluid simulation in Batty et al. 07 (BBB07), Batty and Bridson 08
- Modifications to solve LCP
  - To enforce separating solid boundary condition

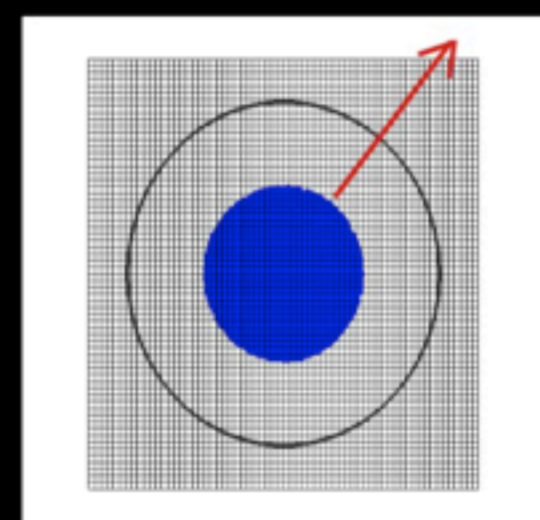
Example

$$(\mathbf{u} - \mathbf{u}_s) \cdot \mathbf{n}_s = 0$$



No wall separating boundary condition

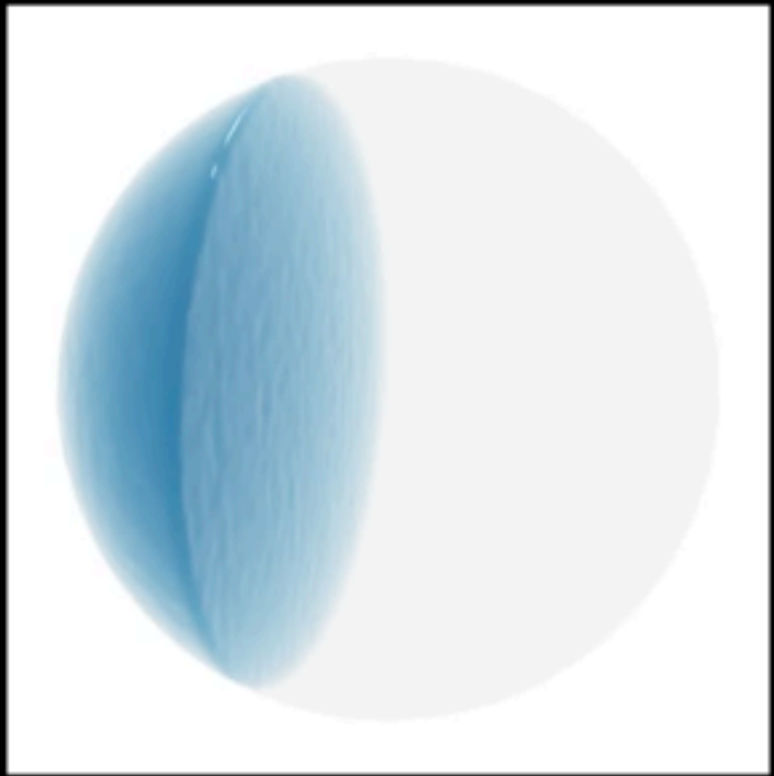
$$(\mathbf{u} - \mathbf{u}_s) \cdot \mathbf{n}_s \geq 0$$



With wall separating boundary condition (Node Base)

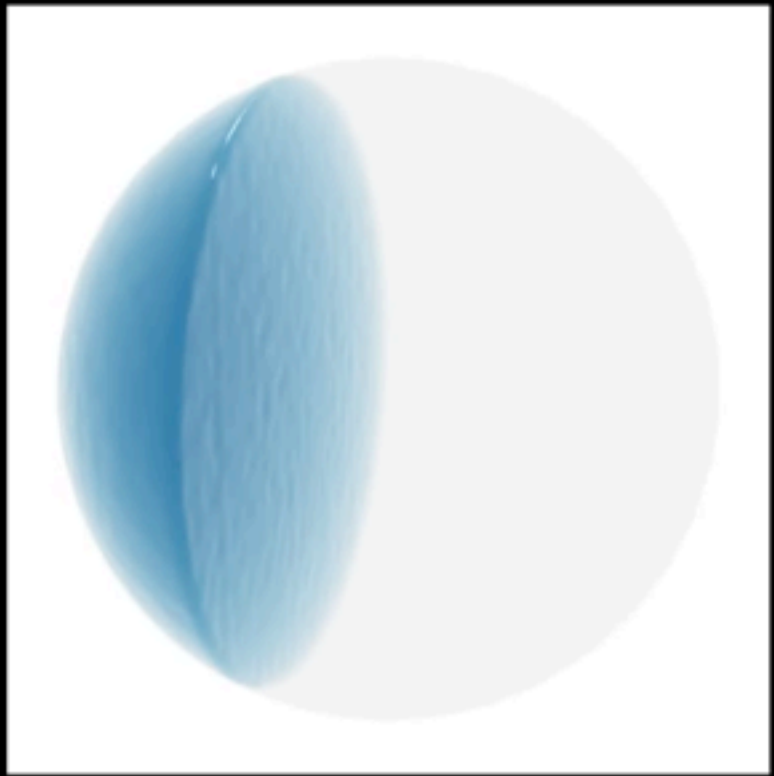
Example

$$(\mathbf{u} - \mathbf{u}_s) \cdot \mathbf{n}_s = 0$$



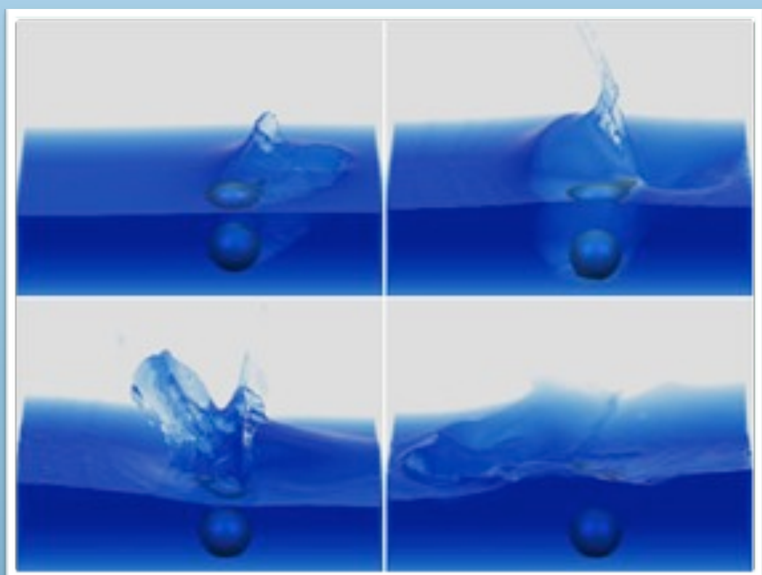
No wall separating boundary condition

$$(\mathbf{u} - \mathbf{u}_s) \cdot \mathbf{n}_s \geq 0$$

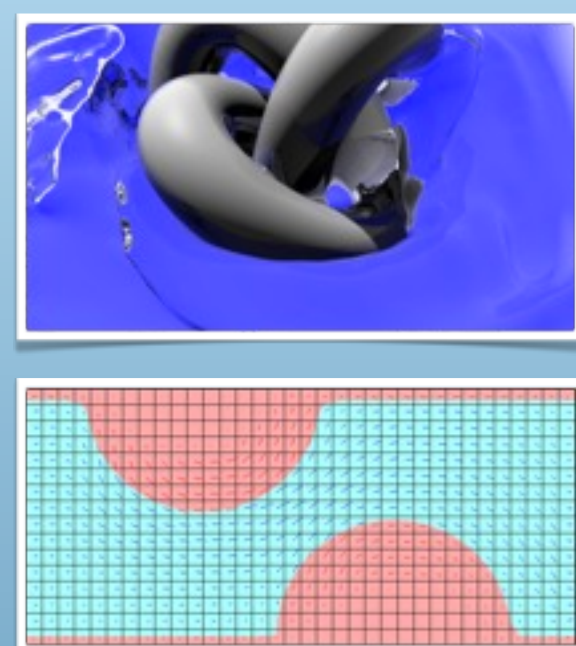


With wall separating boundary condition (Node Base)

Background

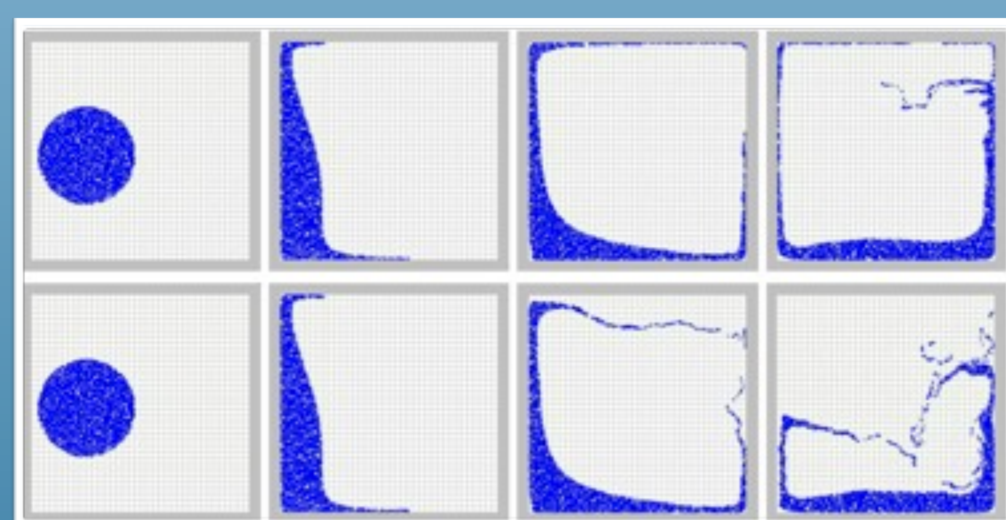


Foster and Fedkiw 01



Houston et al. 03

Rasmussen et al. 04



BBB07

Method

- Inviscid Incompressible Euler Equations

$$\frac{\partial \mathbf{u}}{\partial t} = -(\mathbf{u} \cdot \nabla)\mathbf{u} + \frac{\mathbf{f}}{\rho} - \frac{\nabla p}{\rho}$$

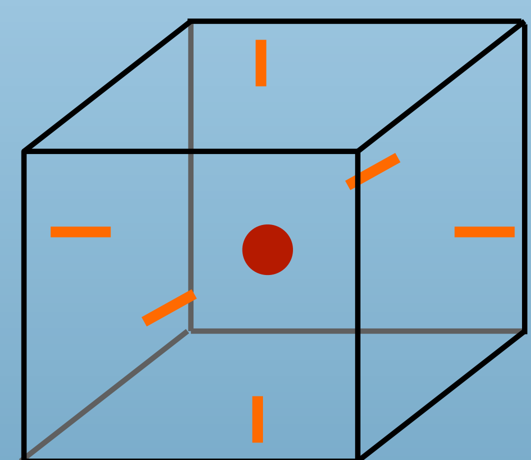
- Subject to $\nabla \cdot \mathbf{u} = 0$

- Inside region $\phi < 0$,

$$\frac{\partial \phi}{\partial t} = -\mathbf{u} \cdot \nabla \phi$$

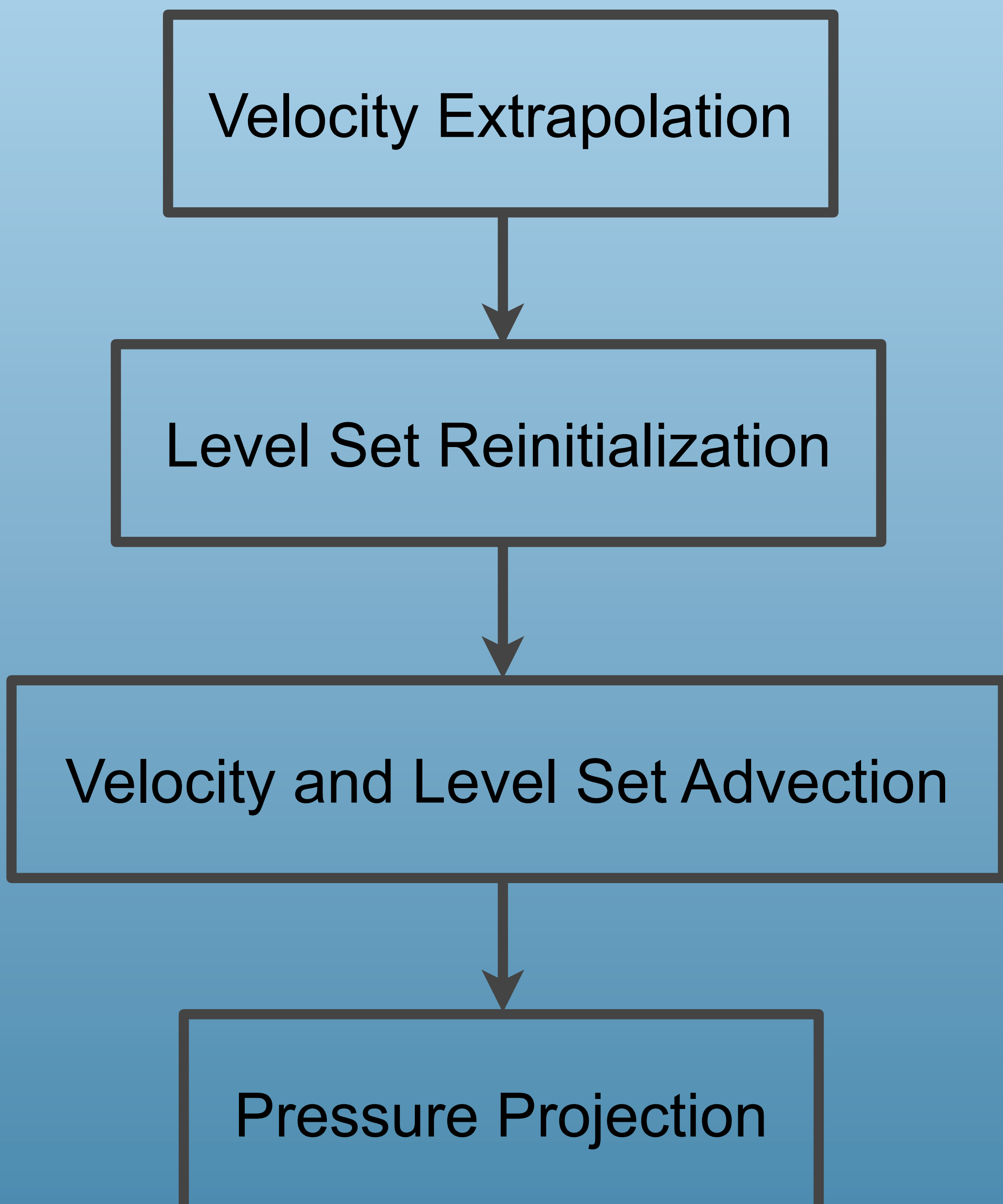
Method

- Discretize to staggered grid as in BBB07



- Cell center
  - Pressure p, level set ϕ, solid fraction V
- Face center
  - Components of velocity $\boldsymbol{u} = [u, v, w]^{\mathrm{T}}$
  - Face center solid fraction V_u, V_v, V_w

Method



- Time integration
  - Standard grid based sim
- Novelty in pressure projection

Pressure Projection

- Let $\mathbf{u}^*$ be the velocity field before pressure projection
- Then

$$\mathbf{u}^{n+1} = \mathbf{u}^* - \frac{\Delta t}{\rho} \nabla p$$

Pressure Projection

- Let $\mathbf{u}^*$ be the velocity field before pressure projection
- Then

$$\mathbf{u}^{n+1} = \mathbf{u}^* - \frac{\Delta t}{\rho} \nabla p$$

- Kinematic energy of the liquid
  - Integrated over the liquid domain

$$\frac{1}{2} \int \mathbf{u}^{n+1} \cdot \mathbf{u}^{n+1} dV$$

  - Take solid fraction and free surface location into account

Pressure Projection

- Let $\mathbf{u}^*$ be the velocity field before pressure projection
- Then

$$\mathbf{u}^{n+1} = \mathbf{u}^* - \frac{\Delta t}{\rho} \nabla p$$

- Kinematic energy of the liquid
  - Integrated over the liquid domain

$$\frac{1}{2} \int \mathbf{u}^{n+1} \cdot \mathbf{u}^{n+1} dV$$

  - Take solid fraction and free surface location into account
- Pressure p found by minimizing the kinetic energy, BBB07
  - Automatically yields divergence free velocity field

Separating Solid Boundary Condition

- Commonly used fluid solver enforces

$$(\mathbf{u} - \mathbf{u}_s) \cdot \mathbf{n}_s = 0$$

  - On the solid boundary
  - Neumann boundary condition
  - Yields linear system that must be solved for p

- For a static ceiling with $\mathbf{u}_s = 0$,
  - Liquid sticks unnaturally

Separating Solid Boundary Condition

- BBB07 propose to enforce

$$(\mathbf{u} - \mathbf{u}_s) \cdot \mathbf{n}_s \geq 0$$

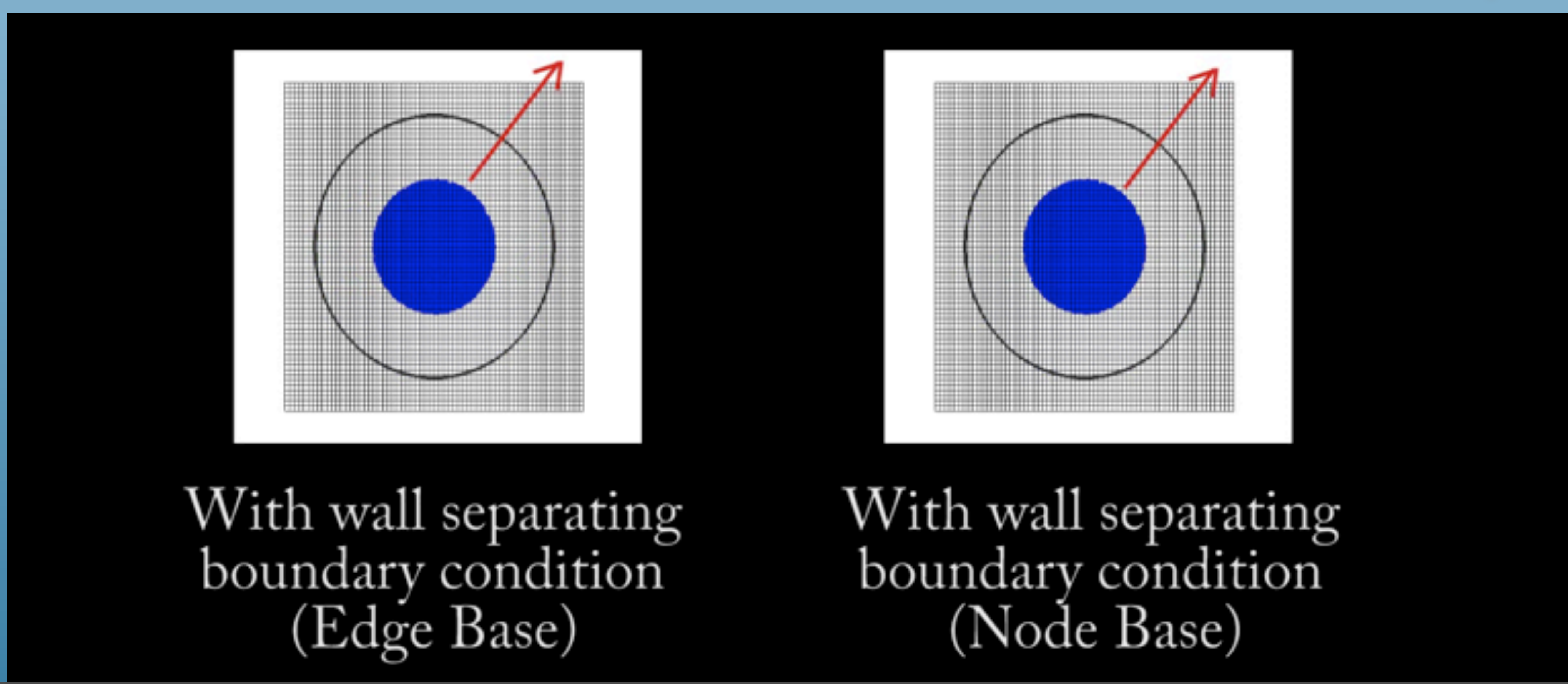
  - If liquid separates from the solid then it becomes a free surface $p = 0$
  - Otherwise, $p > 0$ disallowing suction
- Hence

$$0 \leq p \perp (\mathbf{u} - \mathbf{u}_s) \cdot \mathbf{n}_s$$

  - Linear Complementarity Problem (LCP)
  - Only need to enforce $p \geq 0$, BBB07

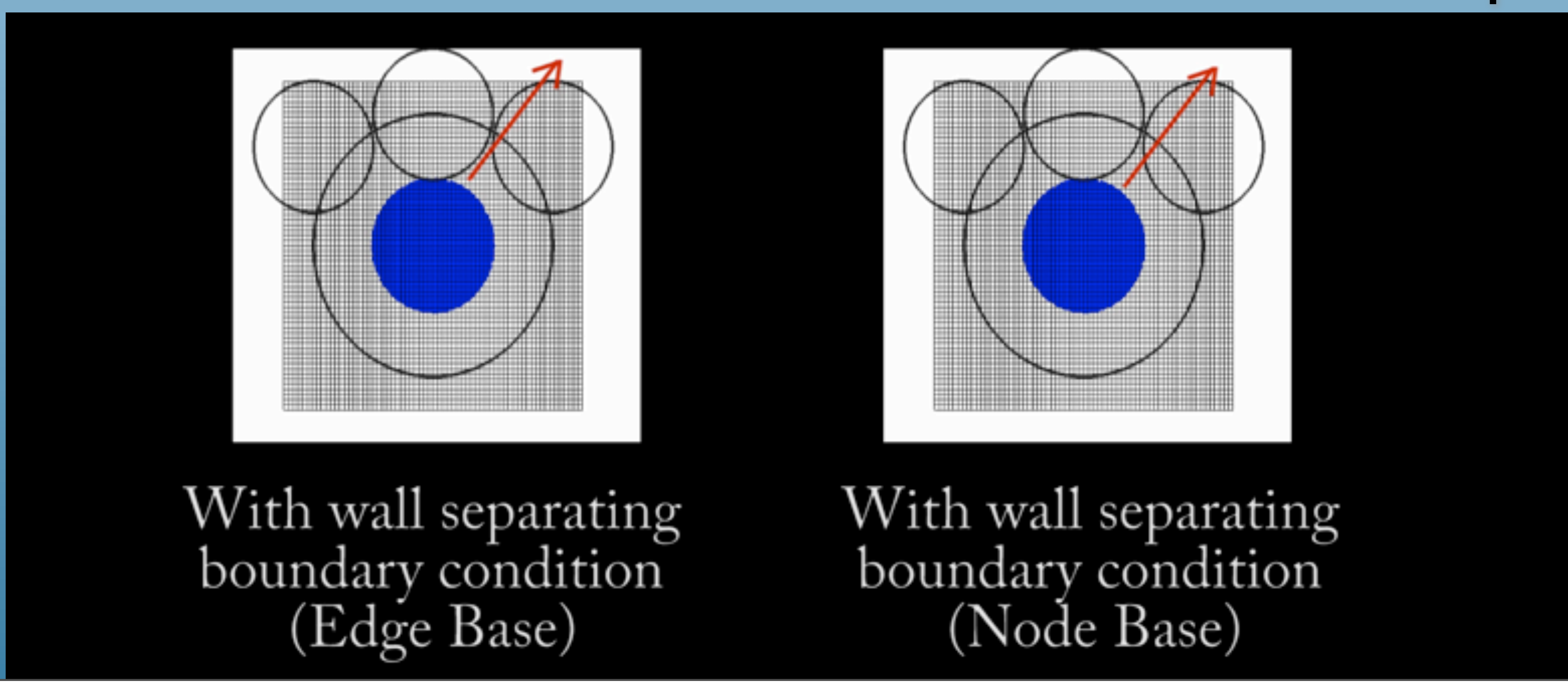
Multigrid LCP solver

- We propose to use a multigrid solver for this
- Important observation
  - Don't need to enforce $p \geq 0$ exactly on solid surface
  - Just need to enforce at solid nodes next to liquid



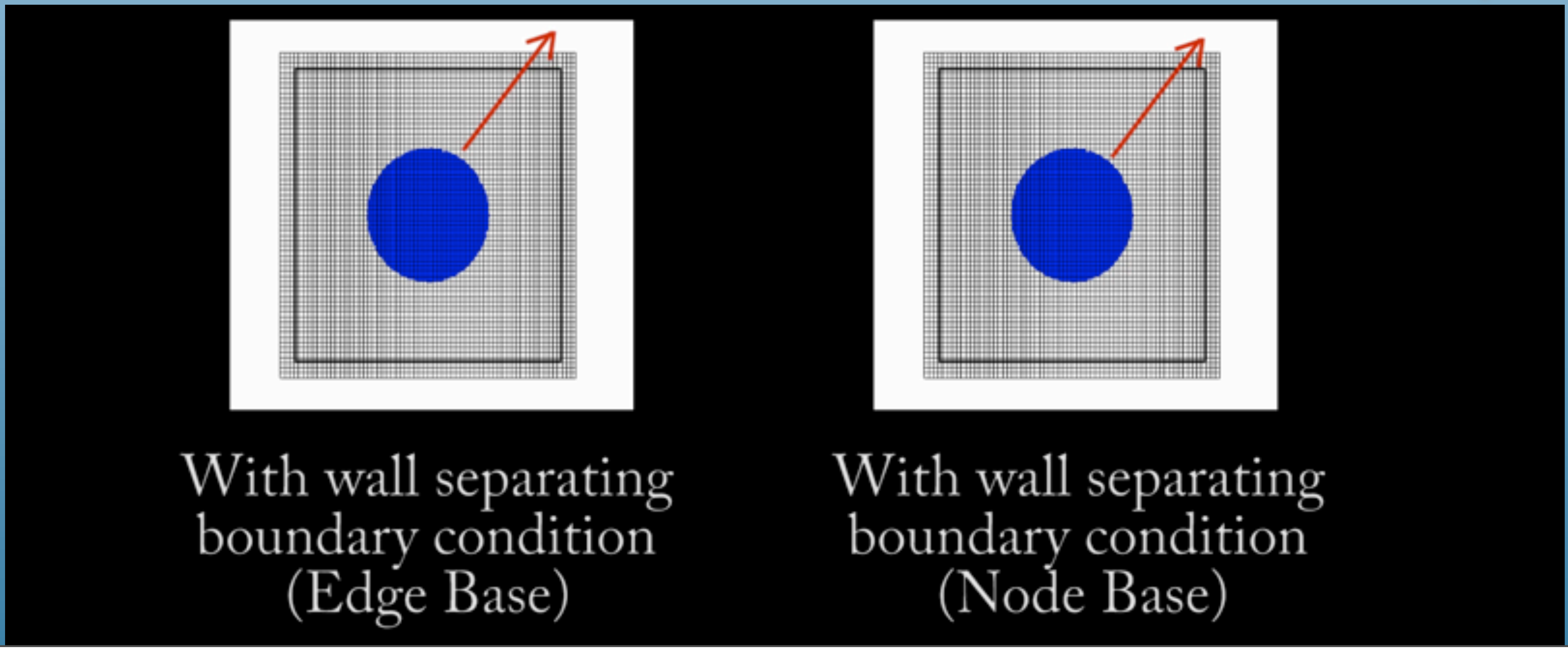
Multigrid LCP solver

- We propose to use a multigrid solver for this
- Important observation
  - Don't need to enforce $p \geq 0$ exactly on solid surface
  - Just need to enforce at solid nodes next to liquid



Multigrid LCP solver

- We propose to use a multigrid solver for this
- Important observation
  - Don't need to enforce $p \geq 0$ exactly on solid surface
  - Just need to enforce at solid nodes next to liquid



Multigrid LCP Solver

- Adapt from MG Solver in Chentanez & Müller 11
- Idea
  - Replace Gauss Seidel with Projected Gauss Seidel

$$Ap = b$$

Multigrid LCP Solver

- Adapt from MG Solver in Chentanez & Müller 11
- Idea
  - Replace Gauss Seidel with Projected Gauss Seidel

$$A_{i,j,k}^{i,j,k} p_{i,j,k} + A_{i,j,k}^{i+1,j,k} p_{i+1,j,k} + A_{i,j,k}^{i-1,j,k} p_{i-1,j,k} + \ldots = b_{i,j,k}$$

Multigrid LCP Solver

- Adapt from MG Solver in Chentanez & Müller 11
- Idea
  - Replace Gauss Seidel with Projected Gauss Seidel

$$p_{i,j,k} = \frac{1}{A_{i,j,k}^{i,j,k}}(b_{i,j,k} - A_{i,j,k}^{i+1,j,k} p_{i+1,j,k} - A_{i,j,k}^{i+1,j,k} p_{i+1,j,k} - \ldots)$$

Multigrid LCP Solver

- Adapt from MG Solver in Chentanez & Müller 11
- Idea
  - Replace Gauss Seidel with Projected Gauss Seidel

$$p_{i,j,k} = \frac{1}{A_{i,j,k}^{i,j,k}}(b_{i,j,k} - A_{i,j,k}^{i+1,j,k} p_{i+1,j,k} - A_{i,j,k}^{i+1,j,k} p_{i+1,j,k} - \ldots)$$

    - Gauss Seidel applies the above equation iteratively

Multigrid LCP Solver

- Adapt from MG Solver in Chentanez & Müller 11
- Idea
  - Replace Gauss Seidel with Projected Gauss Seidel

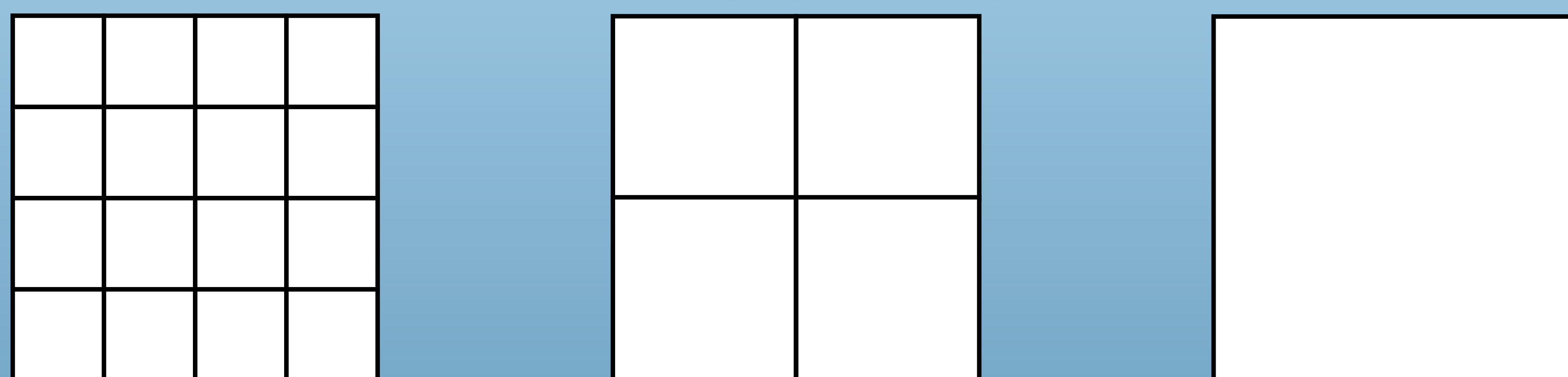
$$p_{i,j,k} = max(p_{\min i,j,k}, \frac{1}{A^{i,j,k}_{i,j,k}}(b_{i,j,k} - A^{i+1,j,k}_{i,j,k} p_{i+1,j,k} - \ldots))$$

    - Projected Gauss Seidel applies the above equation iteratively, where

$$p_{\min i,j,k} = \begin{cases} 0 & \text{if i,j,k is inside a solid} \\ -\infty & \text{otherwise} \end{cases}$$

Multigrid LCP Solver

- Build hierarchy of grids
  - 8-to-1 down sampling (in 3D)

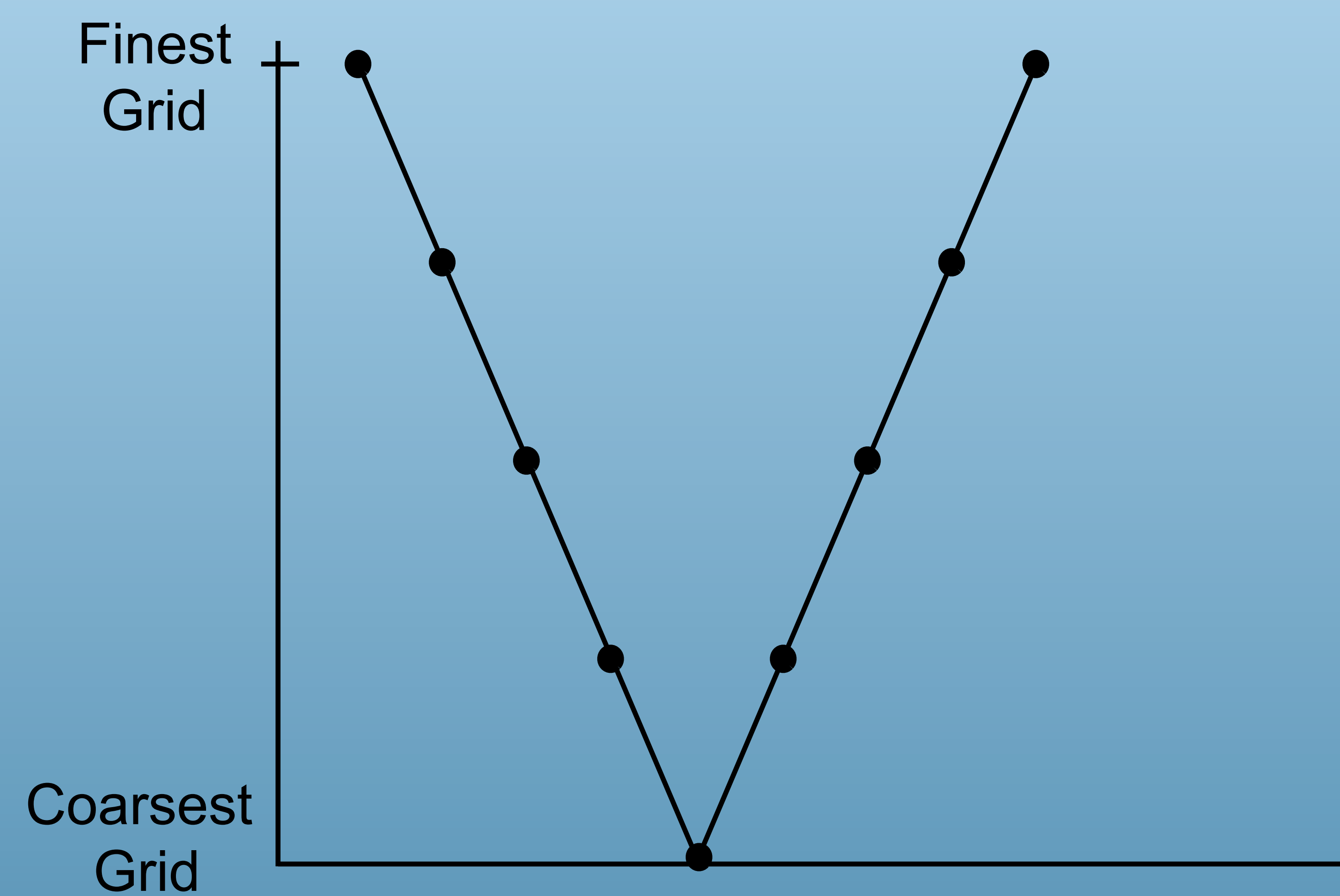


  - Down sampling ϕ specially
    - Preserving air bubbles in a few finest levels, Chentanez & Müller 11

Multigrid LCP Solver

Algorithm 2 V_Cycle(m)

if $m == 1$ **then**
  Solve the linear system, $\mathbf{A}^1 p^1 = b^1$
else
  for $i = 1$ to num_Pre_Sweep **do**
    Smooth(p^m) and enforce p^m_{min} (PRBGS)
  end for
  $r^m = b^l - \mathbf{A} p^m$
  $b^{m-1} = \text{Restrict}(r^m)$
  $p^{m-1} = 0$
  if $m > M - S$ **then**
    $p^{m-1}_{\text{min}} = \text{DownsampleSubtract}(p^m_{\text{min}}, p^m)$
  else
    $p^m_{\text{min}} = -\infty$
  end if
  V_Cycle($m - 1$)
  $p^m = p^m + \text{Prolong}(p^{m-1})$
  for $i = 1$ to num_Post_Sweep **do**
    Smooth (p^m) and enforce p^m_{min} (PRBGS)
  end for
end if



Multigrid LCP Solver

Algorithm 2 V_Cycle(m)

if $m == 1$ **then**
  $\boxed{\text{Solve the linear system, } \mathbf{A}^1 p^1 = b^1}$
else
  for $i = 1$ to num_Pre_Sweep **do**
    Smooth(p^m) and enforce p^m_{min} (PRBGS)
  end for
  $r^m = b^l - \mathbf{A} p^m$
  $b^{m-1} = \text{Restrict}(r^m)$
  $p^{m-1} = 0$
  if $m > M - S$ **then**
    $p^{m-1}_{\text{min}} = \text{DownsampleSubtract}(p^m_{\text{min}}, p^m)$
  else
    $p^m_{\text{min}} = -\infty$
  end if
  V_Cycle($m - 1$)
  $p^m = p^m + \text{Prolong}(p^{m-1})$
  for $i = 1$ to num_Post_Sweep **do**
    Smooth (p^m) and enforce p^m_{min} (PRBGS)
  end for
end if

Base Case

Multigrid LCP Solver

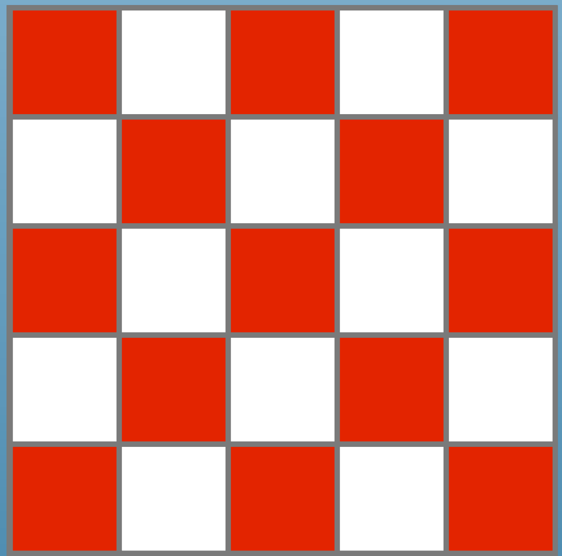
Algorithm 2 V_Cycle(m)

if $m == 1$ **then**
  Solve the linear system, $\mathbf{A}^1 p^1 = b^1$
else
  for $i = 1$ to num_Pre_Sweep **do**
    $\boxed{\text{Smooth}(p^m) \text{ and enforce } p^m_{\text{min}} \text{ (PRBGS)}}$
  end for
  $r^m = b^l - \mathbf{A} p^m$
  $b^{m-1} = \text{Restrict}(r^m)$
  $p^{m-1} = 0$
  if $m > M - S$ **then**
    $p^{m-1}_{\text{min}} = \text{DownsampleSubtract}(p^m_{\text{min}}, p^m)$
  else
    $p^m_{\text{min}} = -\infty$
  end if
  V_Cycle($m - 1$)
  $p^m = p^m + \text{Prolong}(p^{m-1})$
  for $i = 1$ to num_Post_Sweep **do**
    Smooth (p^m) and enforce p^m_{min} (PRBGS)
  end for
end if

Pre-smoothing

$$p_{i,j,k} = max(p_{\min i,j,k}, \frac{1}{A^{i,j,k}_{i,j,k}}(b_{i,j,k} - A^{i+1,j,k}_{i,j,k} p_{i+1,j,k} - \ldots))$$

Projected Red Black Gauss Seidel



Multigrid LCP Solver

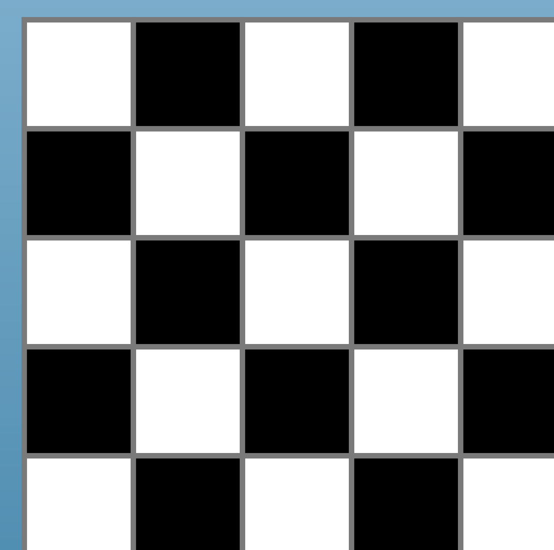
Algorithm 2 V_Cycle(m)

if $m == 1$ **then**
  Solve the linear system, $\mathbf{A}^1 p^1 = b^1$
else
  for $i = 1$ to num_Pre_Sweep **do**
    $\boxed{\text{Smooth}(p^m) \text{ and enforce } p^m_{\text{min}} \text{ (PRBGS)}}$
  end for
  $r^m = b^l - \mathbf{A}p^m$
  $b^{m-1} = \text{Restrict}(r^m)$
  $p^{m-1} = 0$
  if $m > M - S$ **then**
    $p^{m-1}_{\text{min}} = \text{DownsampleSubtract}(p^m_{\text{min}}, p^m)$
  else
    $p^m_{\text{min}} = -\infty$
  end if
  V_Cycle($m-1$)
  $p^m = p^m + \text{Prolong}(p^{m-1})$
  for $i = 1$ to num_Post_Sweep **do**
    Smooth (p^m) and enforce p^m_{min} (PRBGS)
  end for
end if

Pre-smoothing

$$p_{i,j,k} = max(p_{\min\, i,j,k}, \frac{1}{A^{i,j,k}_{i,j,k}}(b_{i,j,k} - A^{i+1,j,k}_{i,j,k} p_{i+1,j,k} - \ldots))$$

Projected Red Black Gauss Seidel



Multigrid LCP Solver

Algorithm 2 V_Cycle(m)

if $m == 1$ **then**
  Solve the linear system, $\mathbf{A}^1 p^1 = b^1$
else
  for $i = 1$ to num_Pre_Sweep **do**
    Smooth(p^m) and enforce p^m_{min} (PRBGS)
  end for
  $r^m = b^l - \mathbf{A} p^m$
  $b^{m-1} = \text{Restrict}(r^m)$
  $p^{m-1} = 0$
  if $m > M - S$ **then**
    $p^{m-1}_{\text{min}} = \text{DownsampleSubtract}(p^m_{\text{min}}, p^m)$
  else
    $p^m_{\text{min}} = -\infty$
  end if
  V_Cycle($m - 1$)
  $p^m = p^m + \text{Prolong}(p^{m-1})$
  for $i = 1$ to num_Post_Sweep **do**
    Smooth (p^m) and enforce p^m_{min} (PRBGS)
  end for
end if

Compute Residual

Multigrid LCP Solver

Algorithm 2 V_Cycle(m)

```
if m == 1 then
    Solve the linear system, A^1 p^1 = b^1
else
    for i = 1 to num_Pre_Sweep do
        Smooth(p^m) and enforce p^m_min (PRBGS)
    end for
    r^m = b^l − A p^m
    b^(m−1) = Restrict(r^m)
    p^(m−1) = 0
    if m > M − S then
        p^(m−1)_min = DownsampleSubtract(p^m_min, p^m)
    else
        p^m_min = −∞
    end if
    V_Cycle(m − 1)
    p^m = p^m + Prolong(p^(m−1))
    for i = 1 to num_Post_Sweep do
        Smooth (p^m) and enforce p^m_min (PRBGS)
    end for
end if
```

Restrict

- Down sampling
- Tri-linear interpolation

Multigrid LCP Solver

Algorithm 2 V_Cycle(m)

```
if m == 1 then
    Solve the linear system, A^1 p^1 = b^1
else
    for i = 1 to num_Pre_Sweep do
        Smooth(p^m) and enforce p^m_min (PRBGS)
    end for
    r^m = b^l − A p^m
    b^{m−1} = Restrict(r^m)
    p^{m−1} = 0
    if m > M − S then
        p^{m−1}_min = DownsampleSubtract(p^m_min, p^m)
    else
        p^m_min = −∞
    end if
    V_Cycle(m − 1)
    p^m = p^m + Prolong(p^{m−1})
    for i = 1 to num_Post_Sweep do
        Smooth (p^m) and enforce p^m_min (PRBGS)
    end for
end if
```

Want to make sure

$$p^m_{i,j,k} + \mathrm{Prolong}(p^{m-1}_{i,j,k}) \geq p^m_{\min\, i,j,k}$$

Guaranteed by

$$p^{m-1}_{\min\, i,j,k} =$$

$$\mathrm{DownsampleSubtract}(p^m_{\min}, p^m)_{i,j,k} =$$

$$\max_{a,b,c \in \{0,1\}} (p^m_{\min\ 2i+a,2j+b,2k+c} - p^m_{2i+a,2j+b,2k+c})$$

Only needed for the finest S levels

Multigrid LCP Solver

Algorithm 2 V_Cycle(m)

if $m == 1$ **then**
    Solve the linear system, $\mathbf{A}^1 p^1 = b^1$
else
    for $i = 1$ to num_Pre_Sweep **do**
        Smooth(p^m) and enforce p^m_{min} (PRBGS)
    end for
    $r^m = b^l - \mathbf{A} p^m$
    $b^{m-1} = \text{Restrict}(r^m)$
    $p^{m-1} = 0$
    if $m > M - S$ **then**
        $p^{m-1}_{\text{min}} = \text{DownsampleSubtract}(p^m_{\text{min}}, p^m)$
    else
        $p^m_{\text{min}} = -\infty$
    end if
    V_Cycle($m - 1$)
    $p^m = p^m + \text{Prolong}(p^{m-1})$
    for $i = 1$ to num_Post_Sweep **do**
        Smooth (p^m) and enforce p^m_{min} (PRBGS)
    end for
end if

Recursive to solve for p

Multigrid LCP Solver

Algorithm 2 V_Cycle(m)

if $m == 1$ **then**
    Solve the linear system, $\mathbf{A}^1 p^1 = b^1$
else
    for $i = 1$ to num_Pre_Sweep **do**
        Smooth(p^m) and enforce p^m_{min} (PRBGS)
    end for
    $r^m = b^l - \mathbf{A}p^m$
    $b^{m-1} = \text{Restrict}(r^m)$
    $p^{m-1} = 0$
    if $m > M - S$ **then**
        $p^{m-1}_{\text{min}} = \text{DownsampleSubtract}(p^m_{\text{min}}, p^m)$
    else
        $p^m_{\text{min}} = -\infty$
    end if
    V_Cycle($m - 1$)
    $\boxed{p^m = p^m + \text{Prolong}(p^{m-1})}$
    for $i = 1$ to num_Post_Sweep **do**
        Smooth (p^m) and enforce p^m_{min} (PRBGS)
    end for
end if

Prolong
- Up sampling
- Tri-linear interpolation

Multigrid LCP Solver

Algorithm 2 V_Cycle(m)

if $m == 1$ **then**
  Solve the linear system, $\mathbf{A}^1 p^1 = b^1$
else
  for $i = 1$ to num_Pre_Sweep **do**
    Smooth(p^m) and enforce p^m_{min} (PRBGS)
  end for
  $r^m = b^l - \mathbf{A} p^m$
  $b^{m-1} = \text{Restrict}(r^m)$
  $p^{m-1} = 0$
  if $m > M - S$ **then**
    $p^{m-1}_{\text{min}} = \text{DownsampleSubtract}(p^m_{\text{min}}, p^m)$
  else
    $p^m_{\text{min}} = -\infty$
  end if
  V_Cycle($m - 1$)
  $p^m = p^m + \text{Prolong}(p^{m-1})$
  for $i = 1$ to num_Post_Sweep **do**
    Smooth (p^m) and enforce p^m_{min} (PRBGS)
  end for
end if

Post Smooth

Multigrid LCP Solver

Algorithm 2 V_Cycle(m)

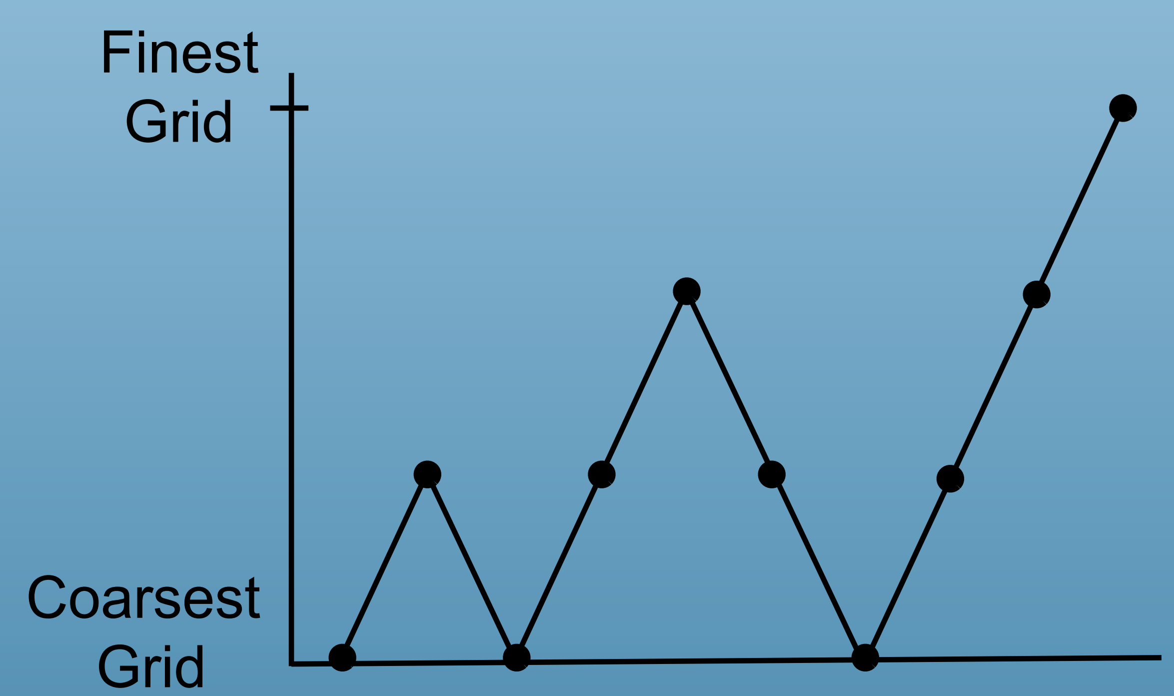
if $m == 1$ **then**
  Solve the linear system, $\mathbf{A}^1 p^1 = b^1$
else
  for $i = 1$ to num_Pre_Sweep **do**
    Smooth(p^m) and $\boxed{\text{enforce } p^m_{\text{min}} \text{ (PRBGS)}}$
  end for
  $r^m = b^l - \mathbf{A} p^m$
  $b^{m-1} = \text{Restrict}(r^m)$
  $p^{m-1} = 0$
  if $m > M - S$ **then**
    $p^{m-1}_{\text{min}} = \text{DownsampleSubtract}(p^m_{\text{min}}, p^m)$
  else
    $p^m_{\text{min}} = -\infty$
  end if
  V_Cycle($m - 1$)
  $p^m = p^m + \text{Prolong}(p^{m-1})$
  for $i = 1$ to num_Post_Sweep **do**
    Smooth (p^m) and $\boxed{\text{enforce } p^m_{\text{min}} \text{ (PRBGS)}}$
  end for
end if

Differences from traditional Multigrid

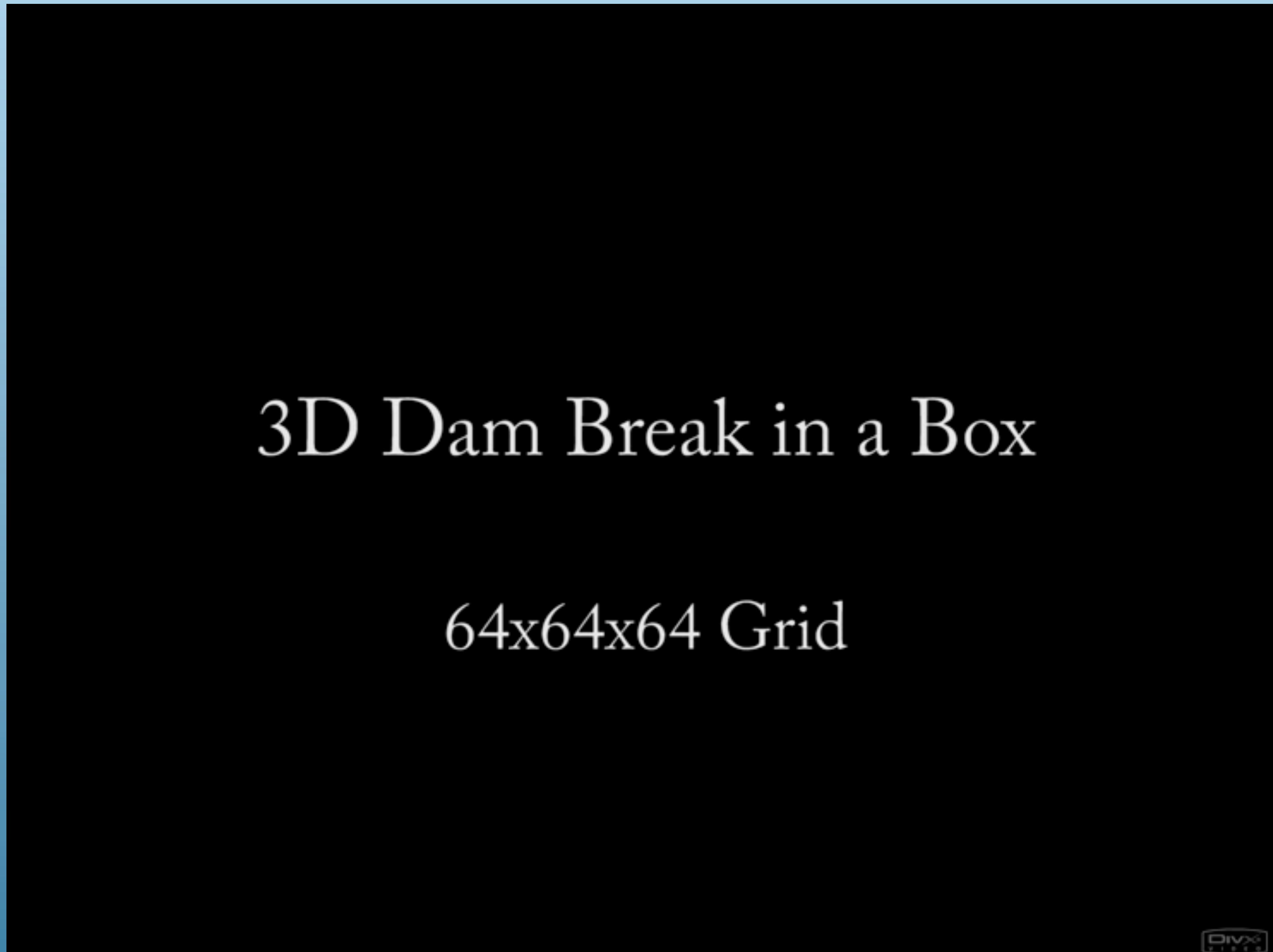
Multigrid LCP Solver

Algorithm 3 Full_Cycle()

$p^{\text{tmp}} = p^M$
Compute p^M_{min}
$p^M_{\text{min}} -= p^M$
$r^M = b^M - \mathbf{A}p^M$
for $m = M-1$ down to 1 **do**
  $r^m = \text{Restrict}(r^{m+1})$
  if $m \geq M - S$ **then**
    $p^m_{\text{min}} = \text{DownsampleSubtract}(p^{m+1}_{\text{min}}, 0)$
  else
    $p^m_{\text{min}} = -\infty$
  end if
end for
$b^1 = r^1$
Solve the linear system, $\mathbf{A}^1 p^1 = b^1$
for $m = 2$ to M **do**
  $p^m = \text{Prolong}(p^{m-1})$
  $b^m = r^m$
  V_Cycle(m)
end for
$p^M = p^{\text{tmp}} + p^M$



Results



Results

- Timing in ms, done in GTX480

Case	Res	No LCP	LCP	% Diff
BallBox	64^3	19.00	21.26	11.89
DambreakBox	64^3	18.89	21.17	12.07
RotatedBox	128^3	109.78	122.97	12.01
DambreakSphere	128^3	109.67	122.58	11.77

- No more than 12% slower than multigrid w/o LCP
  - MG faster than CG about 13X, CM11
  - Expected to be much faster than BBB07
    - Because the solver used was much slower than CG

Discussions

- Only one way solid-liquid coupling is currently supported
- Two-way solid-liquid such as by incorporating Robinson-Mosher et al. 08
  – Will be challenging and interesting future work

Thank you for your attention!

Solving LCP

- BBB07 formulate as quadratic programming (QP
  - Used PATH solver for it
  - Slow, feasible only for small 2D domain
- Narian et al. 10
  - Solve LCP resulting from sand simulation
  - Use conjugate gradient - liked QP solver