Stability of an Embedded Mesh Method for Coupling Lagrangian and ALE Finite Element Models

M. A. Puso, E. J. Kokko, B. T. Liu, B. Simpkins

December 12, 2013

AHPCRC Workshop on Evolving Discontinuities
Stanford, CA, United States
December 3, 2013 through December 5, 2013
Stability of an Embedded Mesh Method for Coupling Lagrangian and ALE Finite Element Models

AHPCRC Workshop on Evolving Discontinuities
Stanford University, Dec 3 2013

Mike Puso, Ed Kokko, Ben Liu, Bryan Simpkins
Methods Development Group, Computational Geosciences Group
Lawrence Livermore National Laboratory
Livermore CA

LLNL-PROC-647639

Work performed under the HPC Software Application Institute on Blast Protection for Platforms and Personnel under funding from the DoD HPC Modernization Program

Lawrence Livermore National Laboratory
Outline

- What is embedded mesh?
- Previous works
- Two Step ALE approach based on time splitting
  - Lagrange Step
    - Mesh Coupling: *Focus on following stability issues*
      - Stability of the multiplier space
      - Condition number
      - Stable time step
  - Advection Remap Step
- Results
  - Verifications, blast, failure etc.
- Summary
Embedded Mesh technique couples overlapping meshes

- **Background Mesh:** Eulerian, ALE, Lagrange

- **Foreground Mesh:** Solid or Shell or Membrane
  - Simple: Avoids construction of *body fitted mesh*
  - Robust: Minimizes mesh tangling
For Example
Many Previous Works: to name a few

- Existing *Embedded Mesh* methods
  - *CEL method* (W.F. Noh, 1964)
  - *Overset grid methods* (W.D. Henshaw 2006)
  - *Zapotec material insertion method* (Bessette 2002)
    - Sandia code couples CTH and Pronto
  - *LS-Dyna, ABAQUS* (commercial codes)
  - *Mortar fictitious domain methods* (Baaijens 2001)
  - *Nitsche’s Method* (Hansbo and Hansbo, 2003; Sanders and Puso 2010)
  - *Ghost Fluid methods* (Fedikew et. al. 1999)
    - *DYSMAS* Gemini-PARADYN (Luton et. al. 2003)
Issues Regarding Different Methods

- Many developed for Eulerian-Finite Volume/Difference method
- Overset methods require auxiliary mesh
- Many require penalties
- Some don’t work for shells
- Some not “stable” and/or “consistent”
- Considered too inaccurate for many applications
New Approaches and Goals

More recent approaches

A discontinuous-Galerkin based immersed boundary method (Lew and Buscagli, 2008)
A stable Lagrange multiplier space for stiff interface conditions … (Bechet, Moes, Wohlmuth, 2009)
Embedded Dirichlet Method… (Gerstenberg and Wall, 2010) (Baiges et al. 2012)
Embedded Mesh… DG method (Sanders and Puso, 2012)
Fictitious domain finite element methods using cut elements… (Burman and Hansbo, 2010)
New Approaches and Goals

More recent approaches

- A discontinuous-Galerkin based immersed boundary method (Lew and Buscaglia, 2008)
- A stable Lagrange multiplier space for stiff interface conditions … (Bechet, Moes, Wohlmuth, 2009)
- Embedded Dirichlet Method… (Gerstenberg and Wall, 2010) (Baiges et al. 2012)
- Embedded Mesh… DG method (Sanders and Puso, 2012)
- Fictitious domain finite element methods using cut elements… (Burman and Hansbo, 2010)

Algorithmic Design Constraints

- Suitable for explicit finite element code i.e. good estimates for stable time step
- Suitable for standard finite elements i.e. use linear hexahedral, shell type elements
- Symmetric formulations
- Avoid penalties
Mesh Coupling Approach

Two basic Lagrange multiplier approaches

- Constraints (multipliers) on foreground mesh (known issues)
- Constraints (multipliers) on background mesh (not as easy)

\[ \int_{\Gamma} \lambda \cdot (\mathbf{v}^b - \mathbf{v}^f) \, d\Gamma = 0 \]

Why use Lagrange multipliers?

- Do a good job enforcing constraint (better than penalty)
- Easier to do get symmetric forms (e.g. with Niche)
- Natural way to incorporate “added mass” as opposed to staggered approach
- Turns out not to drastically affect performance
Constraints defined on foreground mesh

- Use standard Lagrange multipliers

\[ \lambda = \sum_{A=1}^{N} N_A(\xi) \lambda_A \]

\[ \int_{\Gamma} \lambda \cdot (v^b - v^f) \, d\Gamma = 0 \]

\[ E = 1000 \quad E = 1 \]
Many commercial codes apply constraints on foreground mesh
  - Penalized constraints on tracer particles
  - Based on collocation, not integrals
  - “Leak” if $k$ is too small
  - “Lock” if $k$ is too big
Constraints on background mesh

• Edge based constraints
  (Bech et al. 2009, Puso et al. 2012)
• Piecewise constants
  (Embedded Dirichlet, Burman Hansbo 2011; Embedded Mesh, Puso 2012 et al.)
  – Simple
  – Requires stabilization

\[ \lambda_1 \lambda_2 \lambda_3 \lambda_4 \]
Multipliers on background mesh: Beam result

background

foreground

![Graph showing error vs. mesh size for background and foreground dispersions and energies. The graph includes two linear trends: one with a slope of 1.07 and another with a slope of 1.99.](image-url)
Multipliers on background mesh: 2D result

- Option: UCRL
- Option: ichelectorate/Department Additional Information

- Multipliers on background mesh
- E = 50,000, E = 5,000
- P = 1, conforming, foreground, background
Multipliers on background mesh: 2D result

- Slope of Background dispersion: 1.97
- Slope of Background energy: 1.15
- Slope of Foreground dispersion: 1.97
- Slope of Foreground energy: 1.15

Mesh size: 0.15 to 1.50

Error values: 0.0001 to 0.1
Mathematical details of approach

- Stability of multiplier space
  - Determine conditions for LBB stability
- Solution to equations of motion
  - Show that condition number is independent of mesh size
- Stability of time integrator
  - Estimate time stable time step
Stability of multiplier space

- Consider abstract form of BVP:
  \[ a(u_h, v_h) + b(\lambda_h, u_h) = \langle f, v \rangle \]
  \[ b(\mu_h, u_h) - j(\mu_h, \lambda_h) = 0 \]

- e.g elasticity
  \[ a(u_h, v_h) = \int_{\Omega^b/\Omega^f} \nabla v_h^b : C^b \nabla u_h^b \, d\Omega + \int_{\Omega^f} \nabla v_h^f : C^f \nabla u_h^f \, d\Omega \]
  \[ b(\lambda_h, u_h^b - u_h^f) = \int_{\Gamma} \lambda \cdot (u_h^b - u_h^f) \, d\Gamma \]

- Checkerboard mode: constraint force at node \( f_A = 0 \), \( \lambda_K \neq 0 \) a.e.

\[ f_A = \sum_{K \in S_A^c} B_{KA} \lambda_K \quad B_{KA} = \int_{\Gamma_K} \phi_A \, d\Gamma \]

- Add stabilization term \( j \): determine conditions for stability

\[ \mathcal{B}[(u_h, \lambda_h), (v_h, \mu_h)] = a(u_h, v_h) + b(\lambda_h, v_h) + b(\mu_h, u_h) - j(\lambda_h, \mu_h) \]

\[ \sup_{\{v_h, \mu_h\} \in \mathcal{W}_h} \frac{\mathcal{B}[(u_h, \lambda_h), (v_h, \mu_h)]}{\|v_h\|_{1,\Omega} + \|\mu\|_{-1/2,\Omega,\Gamma}} \geq \|u_h\|_{1,\Omega} + \|\lambda\|_{-1/2,\Omega,\Gamma} \]

- Choosing appropriate pair \((v_h, \mu_h)\) satisfies inequality

\[ v_h^b = u_h^b + \alpha f_h \quad \text{where} \quad f_h = \sum_{A \in N^c} \phi_A f_A \quad v_h^f = u_h^f \quad \mu_h = -\lambda_h \]
Stability of multiplier space

- **Stability condition**
  \[ \int_{\Gamma} \lambda_h f_h \, d\Gamma + j(\lambda_h, \lambda_h) \geq c_{min} \| \lambda_h \|^2_{-1/2,h} \]

- **Stabilization form**
  \[ j(\lambda_h, \lambda_h) = \sum_{f \in \mathcal{F}_c} \gamma_f h^2 (\lambda_{K_{f1}} - \lambda_{K_{f2}}) \]

- **Recall from before**
  \[ f_h = \sum_{A \in \mathcal{N}_c} \phi_A f_A \quad f_A = \sum_{K \in \mathcal{S}_A^c} B_{KA} \lambda_K \quad B_{KA} = \int_{\Gamma_K} \phi_A \, d\Gamma \]

- **Mesh dependent \( L_2 \) norm**
  \[ \| \lambda \|^2_{-1/2,h} = h \int_{\Gamma} \lambda_h^2 \, d\Gamma \]

- **Compute ratio, with substitution**
  \[
  \frac{\int_{\Gamma} \lambda_h f_h \, d\Gamma + j(\lambda_h, \lambda_h)}{\| \lambda_h \|^2_{-1/2,h}} = \sum_{A \in \mathcal{N}_c} \left[ \frac{1}{4} \left( \sum_{K \in \mathcal{S}_A^c} B_{KA} \lambda_K \right)^2 + \sum_{f \in \mathcal{F}_A^c} \gamma_f h^2 (\lambda_{K_{f1}} - \lambda_{K_{f2}})^2 \right] \geq c_{min}
  \]
Solution to EOM: matrix decomposition

- Central difference equations of motion

\[ M^b(v^b_{n+1/2} - v^b_{n-1/2})/\Delta t + B^b T \lambda = f^b_{n-1} \]
\[ M^f(v^f_{n+1/2} - v^f_{n-1/2})/\Delta t + B^f T \lambda = f^f_{n-1} \]
\[ B^b v^b_{n+1/2} + B^f v^f_{n+1/2} - J/\Delta t \lambda = 0 \]

- Exploiting diagonal mass, solve for \( v^b_{n+1/2} \) and \( v^f_{n+1/2} \) in terms of \( \lambda \) e.g.

\[ v^b_{n+1/2} = -\Delta t M^b^{-1} B^b T \lambda + M^b^{-1} \tilde{f}^b \]

- Substituting into bottom row yields following decomposition

\[ H \lambda = f \]
\[ H = \Delta t^2 (B^b M^b^{-1} B^b T + B^f M^f^{-1} B^f T) + J \]
\[ f = \Delta t B^b M^b^{-1} \tilde{f}^b + \Delta t B^f M^f^{-1} \tilde{f}^f \]

- Use CG solver to find \( \lambda \)
Solution to EOM: matrix decomposition

- Central difference equations of motion

\[ \begin{bmatrix} M^b & 0 & B^b \, T \\ 0 & M^f & B^f \, T \\ B^b & B^f & -J/\Delta t^2 \end{bmatrix} \begin{bmatrix} v^b_{n+1/2} \\ v^f_{n+1/2} \\ \lambda \Delta t \end{bmatrix} = \begin{bmatrix} f^b \Delta t + M^b v^b_{n-1/2} \\ f^f \Delta t + M^f v^f_{n-1/2} \\ 0 \end{bmatrix} = \begin{bmatrix} \tilde{f}^b \\ \tilde{f}^f \\ 0 \end{bmatrix} \]

- Exploiting diagonal mass, solve for \( v^b_{n+1/2} \) and \( v^f_{n+1/2} \) in terms of \( \lambda \) e.g.

\[ v^b_{n+1/2} = -\Delta t M^{-1} B^b \, T \lambda + M^{-1} \tilde{f}^b \]

- Substituting into bottom row yields following decomposition

\[ H \lambda = f \]

\[ H = \Delta t^2 (B^b M^{-1} B^b \, T + B^f M^{-1} B^f \, T) + J \]

\[ f = \Delta t B^b M^{-1} \tilde{f}^b + \Delta t B^f M^{-1} \tilde{f}^f \]

- Use CG solver to find \( \lambda \)
Solution to EOM: condition number

- Condition number determines rate of convergence. Recall that

\[ H = \Delta t^2 (B^b M^{b-1} B^{bT} + B^f M^{f-1} B^{fT}) + J \]

- Stable time step is based on mesh size \( h \)

\[ \Delta t^2 \| M^{-1} \| \propto \left( \frac{h}{c} \right)^2 \left( \frac{1}{\rho h^3} \right) \propto \frac{1}{h} \implies \frac{c_{\text{min}}}{h} \leq \Delta t^2 \| M^{-1} \| \leq \frac{c_{\text{max}}}{h} \]

\[ \lambda_{\text{min}}^{eig}(H) = \min_{\lambda^T \lambda = 1} \lambda^T H \lambda \]

\[ \lambda_{\text{max}}^{eig}(H) = \max_{\lambda^T \lambda = 1} \lambda^T H \lambda \]

\[ \geq \min_{\lambda^T \lambda = 1} \frac{c_{\text{min}}}{h} \lambda^T B^b B^{bT} \lambda + \lambda^T J \lambda \]

\[ \geq c_{\text{min}} \| \lambda_h \|_{-1/2,h}^2 \]

\[ \leq \max_{\lambda^T \lambda = 1} \frac{c_{\text{max}}}{h} \lambda^T B^b B^{bT} \lambda + \lambda^T J \lambda \]

\[ \leq c_{\text{max}} \| \lambda_h \|_{-1/2,h}^2 \]

- The condition number does not change with mesh refinement

\[ s_{\text{con}} = \frac{\lambda_{\text{max}}^{eig}}{\lambda_{\text{min}}^{eig}} = \frac{c_{\text{max}}}{c_{\text{min}}} \]
Stable Time Step (WIP)

- Compute spectral decomposition of $J$
  
  \[ J = \begin{bmatrix} \Phi_{md}^T, \Phi_{ker}^T \end{bmatrix} \begin{bmatrix} \Psi & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \Phi_{md} \\ \Phi_{ker} \end{bmatrix} \]

  where $\Psi = \text{diag}(\psi_1, \ldots \psi_{md})$ and $\Phi_{md} \in \mathbb{R}^{md \times n}$, $\Phi_{ker} \in \mathbb{R}^{(n-md) \times n}$

- Constraints equations are now rewritten

  \[
  \mu^T (B^b v_{n+1/2} + B^f v_{n+1/2}^f - J/\Delta t \lambda) \\
  = [\mu_{md}^T, \mu_{ker}^T] \left( \begin{bmatrix} \Phi_{md} \\ \Phi_{ker} \end{bmatrix} (B^b v_{n+1/2} + B^f v_{n+1/2}^f) - \frac{1}{\Delta t} \begin{bmatrix} \Psi & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \lambda_{md} \\ \lambda_{ker} \end{bmatrix} \right) = 0
  \]

- For sake of analysis, add penalty term to un-damped part $\varepsilon \to 0$

  \[
  \Phi_{md} (B^b v_{n+1/2} + B^f v_{n+1/2}^f) \Delta t - \Psi \lambda_{md} = 0 \\
  \Phi_{ker} (B^b v_{n+1/2} + B^f v_{n+1/2}^f) \Delta t - \varepsilon \lambda_{ker} = 0
  \]
Solving for $\lambda_{m_d}$ and $\lambda_{ker}$ in terms of velocity leads to damped EOM:

$$M \begin{bmatrix} v_{n+1/2}^b \\ v_{n+1/2}^f \end{bmatrix} + C \begin{bmatrix} v_{n+1/2}^b \\ v_{n+1/2}^f \end{bmatrix} = \begin{bmatrix} \tilde{f}_b \\ \tilde{f}_f \end{bmatrix}$$

where the damping matrix $C$ is defined:

$$C = \begin{bmatrix} B^b_T & B^f_T \end{bmatrix} \begin{bmatrix} \Phi_{m_d}^T \Psi^{-1} \Phi_{m_d} + \frac{1}{\varepsilon} \Phi_{ker}^T \Phi_{ker} \end{bmatrix} [B^b, B^f] \Delta t$$
Stable Time Step

- Develop amplitude matrix from the set of equations

\[
d_n = d_{n-1} + v_{n-1/2} \Delta t
\]

\[(M + C)v_{n+1/2} = -Kd_n \Delta t + Mv_{n-1/2}\]

and rewriting

\[
\begin{bmatrix}
  d_n \\
  v_{n+1/2}
\end{bmatrix} =
\begin{bmatrix}
  I & 0 \\
  0 & D
\end{bmatrix}
\begin{bmatrix}
  I & I \Delta t \\
  -M^{-1}K\Delta t & -M^{-1}K\Delta t^2 + I
\end{bmatrix}
\begin{bmatrix}
  d_{n-1} \\
  v_{n-1/2}
\end{bmatrix} = I^d A^u
\begin{bmatrix}
  d_{n} \\
  v_{n-1/2}
\end{bmatrix}
\]

where

\[
D = (I + M^{-1}C)^{-1} \quad \lambda_{\max}^{eig}(D) \leq 1 \quad \Rightarrow \quad \lambda_{\max}^{eig}(I^d) \leq 1
\]

\[
\lambda_{\max}^{eig}(A^u) \leq 1 \quad \text{since} \quad \Delta t < \frac{2}{\omega_{\max}}
\]
Stable Time Step: Maximum Frequency Estimation

- Stable time step based on Max frequency computed from Rayleigh quotient

\[
\omega_{\text{max}}^2 \leq \sup_{u^b,u^f} \frac{u^b T K^b u^b + u^f T K^f u^f}{u^b T M^b u^b + u^f T M^f u^f} \leq \frac{k_{ec}^{\text{bulk}} \sum_A (B_{ec,A} \cdot u^e_A)^2 V_{ec}}{(1/8) \rho_{ec} V_{ec} \sum_B u^e_B \cdot u^e_B} \approx \left( \frac{c}{h} \right)^2_{ec}
\]

- “Cut elements” have no negative effect on time step since mass lumping distributes equal mass to all nodes
Use central difference explicit 2 step ALE approach

- Load step
  - Lagrange step
  - Advection remap step
  - Advance time step $t_n \rightarrow t_{n+1}$

Diagram:
- Solid mesh (foreground)
- Fluid mesh (background)
- $V_e$
- Pressure
- Use central difference explicit 2 step ALE approach

Load step

- Lagrange step (velocity constraints applied)
- Advection remap step
- Advance time step $t_n \rightarrow t_{n+1}$

$V_e$
Use central difference explicit 2 step ALE approach

- Load step
  - Lagrange step
  - Advection remap step (yields volume flux)
- Advance time step $t_n \rightarrow t_{n+1}$

\[
V_e + \Delta V_e^R = \Delta V_e^L = 0
\]
Verification: 2D blast on plate

- Verify Embedded Grid with conforming ALE Model
- C4 blast loading on 5 cm. thick Al plate in air
- Compare embedded grid method to conforming mesh: 3 mesh densities
Parallel Issues

- Use dynamic domain decomposition
  - Build every time step
    - Move data to new decomposition,
    - Do parallel calculation
      - Compute “cut” background cells
      - Do parallel solve
    - Move data back
- Problem took 28~30 iterations for every mesh refinement

Parallel speedup versus number of processors.

Static Decomposition  Dynamic Decomposition
Piston problem: check energy conservation

- Considered 2 piston problems: 3D rectangular cylinder and 2D slanted piston
- Compare energy loss for conforming and embedded meshes

1x1x2 cm cylinder
0.4 cm thick piston
5:1 slant

- 0.7% loss conforming
- 1.2% loss embedded
- 0.27% loss embedded

After 1 million time steps (1 s)
- 3D: 6% and 10% loss for conforming and embedded meshes respectively
- 2D: 1.6% loss for embedded mesh

oscillates at ~300 Hz

5 km/s
Buried mine blast on 3D plate

- Soil
- TNT
- Plate
- Air

Diagram shows a 3D model of a buried mine blast scenario with layers of soil, TNT, and air, along with a plate structure.
Compare embedded grid to conforming

Embedded Grid at 7.24 ms

Conforming ALE Grid at 7.24 ms
Buried mine blast on 3D plate: velocity
Mesh Study: Blast on structural shell element pipe

- Verify Embedded Grid with conforming ALE Model using shell elements
- C4 blast loading on 2 mm thick Al pipe in air
- Compare embedded grid method to conforming mesh: 4 mesh densities
Mesh Study: Blast on structural shell element pipe

- Compare pipe displacement from *finest* mesh
  - Black line: embedded mesh results
  - Grey line: conforming ALE results

-t = 40
-t = 80
-t = 120
-t = 260

~conforming ALE analysis aborts

![Displacement vs. time graph](image)

![Mesh tangle](image)

- tangling
Mesh Study: Blast on structural shell element pipe

slope = 3.92
slope = 2.01
Structural shells: blast on pipe
Pipe Bomb: Fragmentation

Pipe

C4

Air
pipe fragments

void material “under” pipe fragments

superimposed
Summary

- Develop embedded mesh method using piecewise constant Lagrange multipliers with stabilization
- Demonstrate stability and convergence of method
- CG solution of Lagrange multipliers solved on decomposed system
- Condition number of system is independent of mesh refinement
- Time step not affected by embedded mesh
- Reasonable energy conservation
- Incorporate method in 2 step ALE approach with \( r \) adaptivity
- Verify and validate method
Current/Future work

- Continue verification and validation of approach
- Improve parallel scaling
  - Better dynamic domain decomposition
- Provide analysis for estimates of convergence rates
- Currently adding XFEM into foreground mesh
Results good for many cases, consider beam

E = 1000

E = 1
Multipliers on background mesh: 3D result

foreground

background
- Use central difference explicit 2 step ALE approach

![Diagram of ALE implementation with notation: $V_e \equiv \text{Reference volume of fluid (i.e. element volume/density)}$]
- Use central difference explicit 2 step ALE approach

Load step
- Lagrange step
- Advection remap step
- Advance time step $t_n \rightarrow t_{n+1}$
ALE implementation

- Use central difference explicit 2 step ALE approach

Load step
- Lagrange step
- Advection remap step (yields volume flux)
- Advance time step $t_n \rightarrow t_{n+1}$

$$V_e - \Delta V_e^L + \Delta V_e^R$$

$\Delta V_e^L$ $\Delta V_e^R$
### ALE implementation: Multiple Materials

- Young’s Interface Reconstruction to compute PWL approximation of interface

Consider Cell Volume Fractions (VOF)

<table>
<thead>
<tr>
<th></th>
<th>pure air</th>
<th>mixed</th>
<th>pure water</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0 air 0.0 water</td>
<td>0.25 air 0.75 water</td>
<td>0.0 air 1.0 water</td>
<td></td>
</tr>
<tr>
<td>1.0 air 0.0 water</td>
<td>0.50 air 0.50 water</td>
<td>0.0 air 1.0 water</td>
<td></td>
</tr>
<tr>
<td>1.0 air 0.0 water</td>
<td>0.75 air 0.25 water</td>
<td>0.0 air 1.0 water</td>
<td></td>
</tr>
</tbody>
</table>
ALE implementation: Multiple Materials

- Can typically recover a reasonable approximation of interface when smooth

<table>
<thead>
<tr>
<th>Pure Air</th>
<th>Mixed</th>
<th>Pure Water</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0 air</td>
<td>0.75 air</td>
<td>0.25 air</td>
</tr>
<tr>
<td>0.0 water</td>
<td>0.50 water</td>
<td>0.50 water</td>
</tr>
<tr>
<td>1.0 air</td>
<td>0.25 air</td>
<td>0.0 air</td>
</tr>
<tr>
<td>0.0 water</td>
<td>0.75 water</td>
<td>1.0 water</td>
</tr>
</tbody>
</table>
ALE implementation: Young’s Interface Reconstruction

- Compute nodal VOF from cell VOF
- Assume a linear approximation to material VOF and apply Least Squares for 2D case below

\[ VOF(x) \approx ax + by - d \]

\[ E = \sum_{i=1}^{4} [VOF(x_i) - (VOF_i - 0.5)]^2 \]

\[
\begin{array}{cc}
0.6875 & 0.1875 \\
0.8125 & 0.3125 \\
\end{array}
\]

0.50 air
ALE implementation: Young’s Interface Reconstruction

- Advection remap step determines flux volumes of each face $\Delta V_{e}^{f}$
- Interface determines fractions of material in flux volume
ALE implementation: Young’s Interface Reconstruction

- Modification for embedded mesh: interface determined from solid mesh
- No volume fluxes on “void” side
Previous Works

- Existing *Embedded Mesh* methods for *moving meshes*
    - Finite difference fluid with membranes
    - $h_s << h_f$ otherwise leaks i.e., not consistent
    - Enforces constraints “point-wise” between solid mesh fluid mesh
    - $h_s << h_f$ otherwise leaks i.e., not consistent

  e.g. $h_s > h_f$
Previous Works

- Existing *Embedded Mesh* methods for *moving meshes*
  - *Overset grid methods* (W.D. Henshaw 2006)
    - Finite differences/volume for structured grids
    - Momentum/Flux not conserved across boundary
    - Lose symmetry of $[K]u = f$
    - Difficult to implement for shells

Attach conforming overset fluid grid to solid and tie fluid grids

- Red “ghost” points collocated to outside grid
- Green “ghost” points collocated to inside grid
- Ignore force balance at “ghost” points
Previous Works

- Existing *Embedded Mesh* methods for *moving meshes*

  - **Zapotec method** (Bessette 2002)
    - Based on material insertion/force computation
    - Mainly designed for Lagrangian-Eulerian coupling
    - Scheme is not monolithic
    - Cumbersome for Lagrangian shell
    - Requires material insertion everywhere

1. Update Lagrangian body based on current surface loads
2. “Insert” Lagrangian material, velocities, densities etc. into Eulerian cells
3. Update Eulerian velocities
4. Project Eulerian fluid stress onto Lagrangian surface
Previous Works

- **Existing *Embedded Mesh* methods for moving meshes**
  - **Mortar fictitious domain methods** (Baaijens 2001)
    - 2D implementation for finite elements
    - No Leaks, Conserves momentum, Retains symmetry
    - Requires surface integral (difficult, but tractable)
    - Requires solution to system of equations for constraint (bad for explicit!)

Apply surface integral to constrain fluid and solid surface velocities

\[
\int_{\Gamma} \lambda \cdot (v^s - v^f) \, d\Gamma = 0
\]