Reconnection-based Arbitrary-Lagrangian-Eulerian (ReALE) Method with Adaptive Mesh Refinement and Coarsening

LA-UR-13-22256

W. Bo[1] M.J. Shashkov[2] S.K. Sambasivan[1]

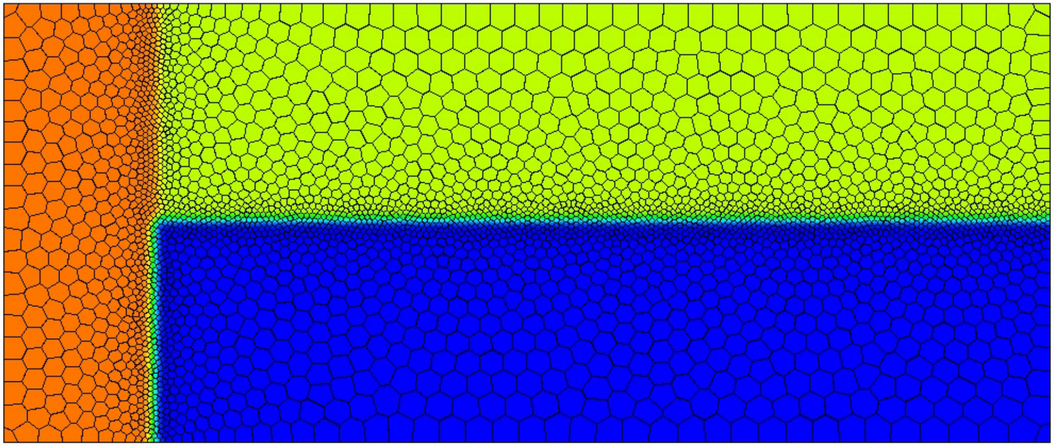
[1]CCS-2, Los Alamos National Lab

[2]XCP-4, Los Alamos National Lab

MULTIMAT 2013

Outline

- Introduction
- H-adaptation
- Numerical results
- Conclusions and perspectives



Introduction

ReALE

- Cell centered Lagrangian formulation of governing equations
- Rezone: Move generators, generate a new mesh using Voronoi tessellation
- Remap: Transfer flow states to the new mesh based on exact intersections of polygonal meshes

Rezone phase in adaptive ReALE

- Move generators in Lagrangian way
- **Global smoothing for the generators**
- **H-adaptation**

[1] Burton, Breil *et.al.*'s talks
[2] P.H. Maire *et.al.* A cell-centered Lagrangian scheme for compressible flow problems, SISC, 2007
[3] R. Loubère *et.al.*, ReALE: A reconnection-based ALE method, JCP, 2010
[4] R. Loubère *et.al.*, An h-adaptive reconnection-based method using Voronoi tessellation, *MULTIMAT'11*
[5] S. Sambasivan, *et.al.*, A finite volume Lagrange approach for computing elasto-plastic deformation of solids on general unstructured grids, JCP, 2013.

Global smoothing

- Relocate generators after the Lagrangian step to improve the shape of the cells
- The new positions of the generators should be close to their Lagrangian positions

Algorithm

1. For each generator i, compute a reference length d_i

j is a neighbor of i

J_j: Jacobian matrix for the Lagrangian step

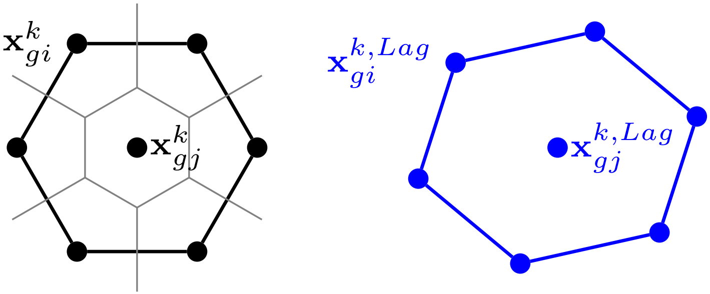
$J_j = \mathbf{U\Sigma V}^*$

$\lambda = |\Sigma|^{1/2}, \hat{J}_j = \lambda \mathbf{UV}^*$

$d_{ji} = ||\hat{J}_j(\mathbf{x}_{gi}^k - \mathbf{x}_{gj}^k) - (\mathbf{x}_{gi}^{k,Lag} - \mathbf{x}_{gj}^{k,Lag})||$

$d_i = \max_j d_{ji}$

2. Lloyd-like iterations under the constrain $||\mathbf{x}_{gi}^{k,\text{new}} - \mathbf{x}_{gi}^{k,Lag}|| \le d_i$



Black: generators at time step k
Blue: generators after the Lagrangian step

Remark

- The algorithm is invariant under translation, rigid body rotation and uniform compression

H-adaptation for a Voronoi mesh

Equi-distribution principle

Given a monitor function $\phi(\mathbf{x}) > 0$ defined on a 2D domain Ω, a partition $\bigcup_{i=1}^{N_g} \Omega_i = \Omega$, equi-distribution principle requires

$$\Phi_i \equiv \int_{\Omega_i} \phi(\mathbf{x}) \mathrm{d}\mathbf{x} = \frac{1}{N_g} \int_{\Omega} \phi(\mathbf{x}) \mathrm{d}\mathbf{x}, \; i = 1 \ldots N_g, \tag{1}$$

where N_g is the number of generators.

H-adaptation

Given bounds $\Phi_{\min}$ and $\Phi_{\max}$, the approximate equi-distribution principle is

$$\Phi_{\min} \leq \Phi_i \leq \Phi_{\max}, \; i = 1 \ldots N_g. \tag{2}$$

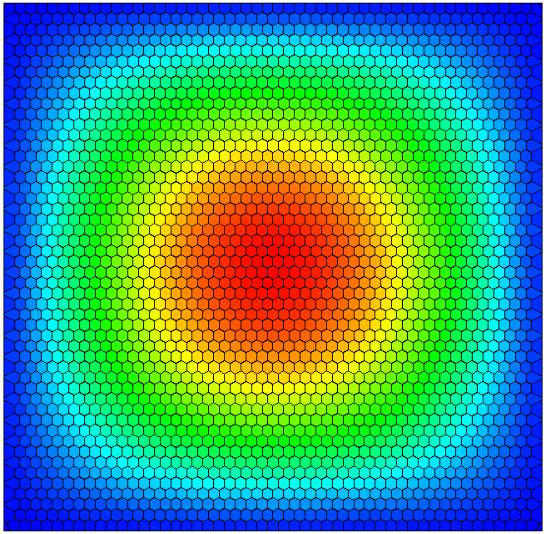
At the end of a hydro step, Φ_i evaluated with the updated monitor function may not be in bounds anymore. We do the following operations such that (2) is maintained for all cells:

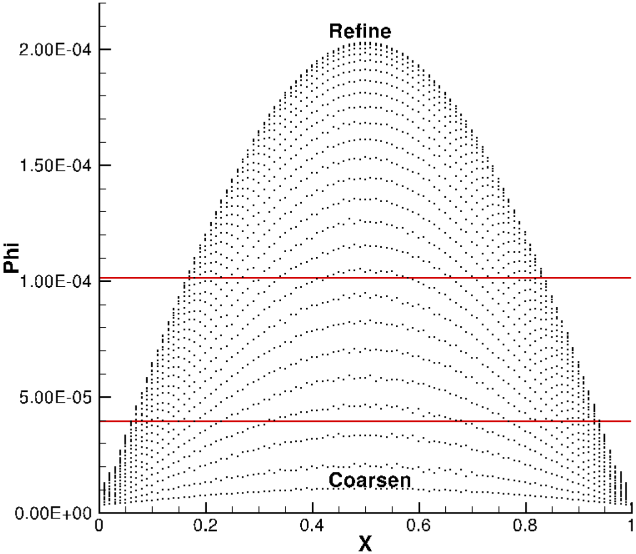
- Insert new generators
- Delete old generators
- Relocate generators

H-adaptation

- Construct a monitor function
- H-adaptation: refinement, local smoothing, coarsening, local smoothing ...

Monitor function $\phi = |\sin(\pi x)\sin(\pi y)| + 0.001$, domain $[0,1] \times [0,1]$. **Left**: The color represents $\Phi_i = \int_{\Omega_i} \phi(\mathbf{x}) d\mathbf{x}$. **Right**: The x coordinates of generators' centroids vs. Φ_i.





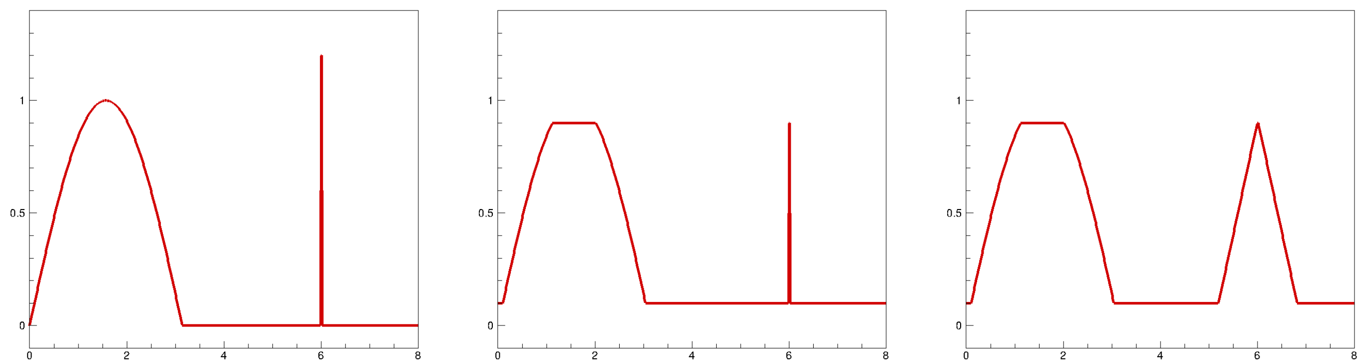
Monitor function

Monitor function

- $\phi(\mathbf{x}) = \max(c_1||\nabla\rho(\mathbf{x})||^2, c_2||\nabla\mathbf{u}(\mathbf{x})||^2, c_3||\nabla E(\mathbf{x})||^2)$. ρ density, $\mathbf{u}$ velocity, E total energy
- Avoid too large or too small cells

$$\tilde{\phi}(\mathbf{x}) \leftarrow \min\left(\max\left(\phi(\mathbf{x}), \phi_{\min}\right), \phi_{\max}\right)$$

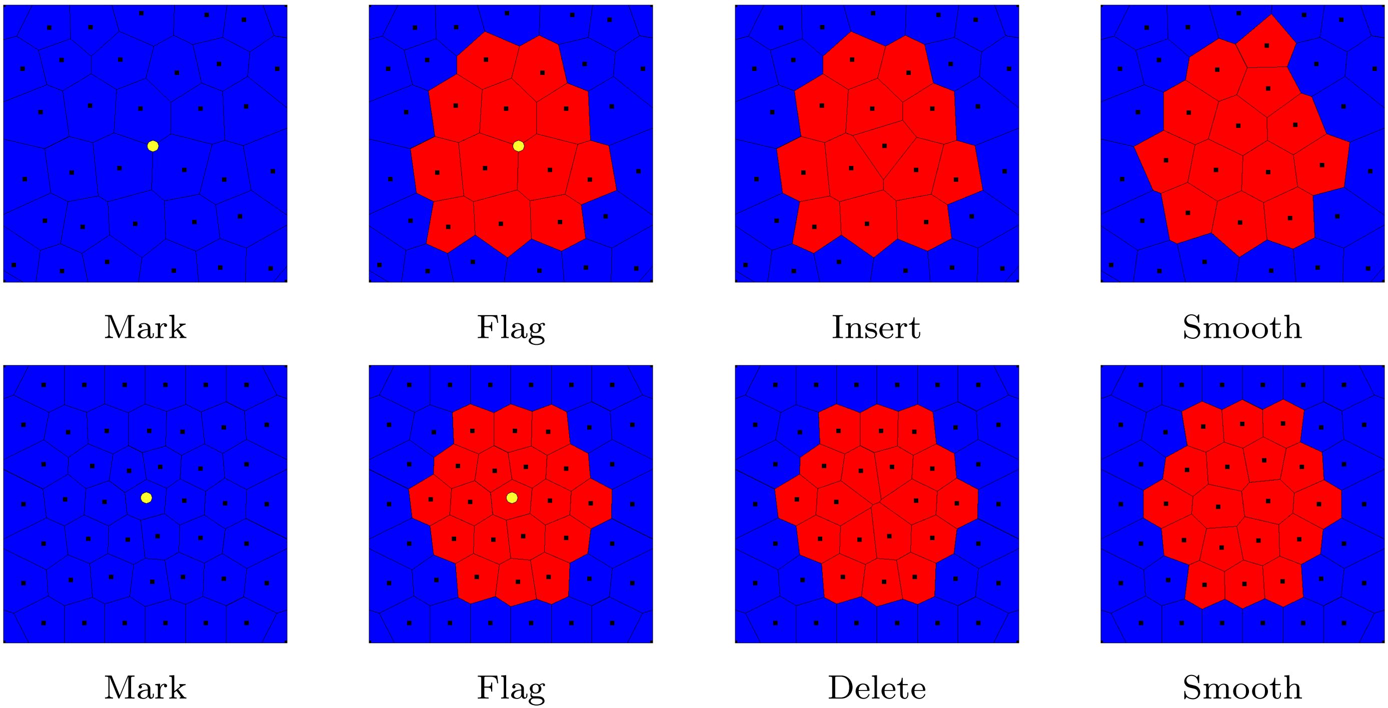
- $c_1, c_2, c_3, \phi_{\min}, \phi_{\max}$ are constants which depend on problems
- To avoid fast change of mesh size, $\tilde{\phi}(\mathbf{x})$ is smoothed. The smoothed monitor function $\hat{\phi}(\mathbf{x})$ satisfies $||\nabla\hat{\phi}(\mathbf{x})|| < 1$ almost everywhere



ϕ $\qquad$ $\tilde{\phi}(\mathbf{x})$ $\qquad$ $\hat{\phi}(\mathbf{x})$

Refinement and coarsening

- Mark the vertex p for insertion if $\max_{i \in \mathcal{P}(p)} \Phi_i > \Phi_{\text{max}}$ and the generators $i : \forall i \in \mathcal{P}(p)$ is not flagged
- Mark the generator i for deletion if $\Phi_i < \Phi_{\text{min}}$ and i is not flagged
- Generators near the inserted/deleted generators are locally smoothed



Local smoothing

Q. Du *et.al.* Centroidal Voronoi Tessellations: Applications and Algorithms, SIAM Review, 1999

Algorithm: Lloyd-like iterations

Given $\{\mathbf{x}_{gi}^0\}$, $\Omega = \bigcup \Omega_i^0$

for n=0:nmax **do**

for $i : \mathbf{x}_{gi}^n$ is flagged **do**

$\mathbf{x}_{ci}^n = \int_{\Omega_i^n} \mathbf{x} \mathrm{d}x / \int_{\Omega_i^n} \mathrm{d}x$

$\mathbf{x}_{mi}^n = \int_{\Omega_i^n} \phi(\mathbf{x})^2 \mathbf{x} \mathrm{d}x / \int_{\Omega_i^n} \phi(\mathbf{x})^2 \mathrm{d}x$

$\omega_i^n = \min(||\mathbf{x}_{gi}^n - \mathbf{x}_{ci}^n|| / ||\mathbf{x}_{mi}^n - \mathbf{x}_{ci}^n||, 1)$

$\mathbf{x}_{di}^n = \omega_i^n \mathbf{x}_{mi}^n + (1 - \omega_i^n) \mathbf{x}_{ci}^n$

Relocate generator i: $\mathbf{x}_{gi}^{n+1} = \mathbf{x}_{di}^n$

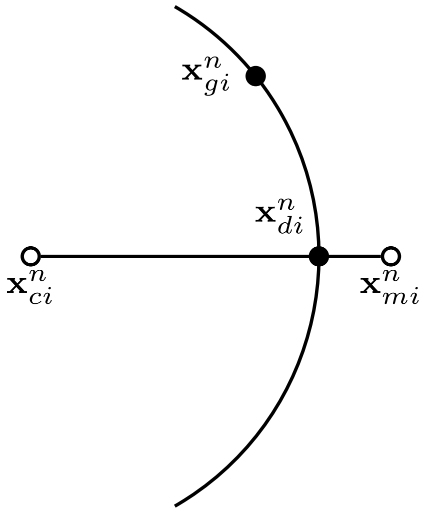
end for

Generate Voronoi mesh $\Omega = \bigcup \Omega_i^{n+1}$

$\mathcal{E}^{n+1} = \sum_i \int_{\Omega_i^{n+1}} ||\mathbf{x} - \mathbf{x}_{gi}^{n+1}||^2 \mathrm{d}\mathbf{x}$

exit the loop if $|\mathcal{E}^{n+1} - \mathcal{E}^n| / \mathcal{E}^n < \epsilon$

end for



Remark

- $\mathcal{E}^n$ is decreasing monotonically

Flow chart of h-adaptation

for n=1:nmax **do**
- **for** vertex p **do** ▷ Mark for refinement, flag generators
  - Mark p if $\max_{i\in\mathcal{P}(p)} \Phi_i > \Phi_{\max}$ and $i : \forall i \in \mathcal{P}(p)$ is not flagged
  - Flag the nearby generators of p
- **end for**
- **for** vertex p **do** ▷ Refinement
  - Insert a generator in p if p is marked
- **end for**
- Local smoothing ▷ Local smoothing
- **for** generator i **do** ▷ Mark for coarsening, flag generators
  - Mark the generator i if $\Phi_i < \Phi_{\min}$ and i is not flagged
  - Flag the nearby generators of i
- **end for**
- **for** generator i **do** ▷ Coarsening
  - Delete the generator i if i is marked
- **end for**
- Local smoothing ▷ Local smoothing
- Break the loop if no generators are inserted or deleted

end for

Numerical results

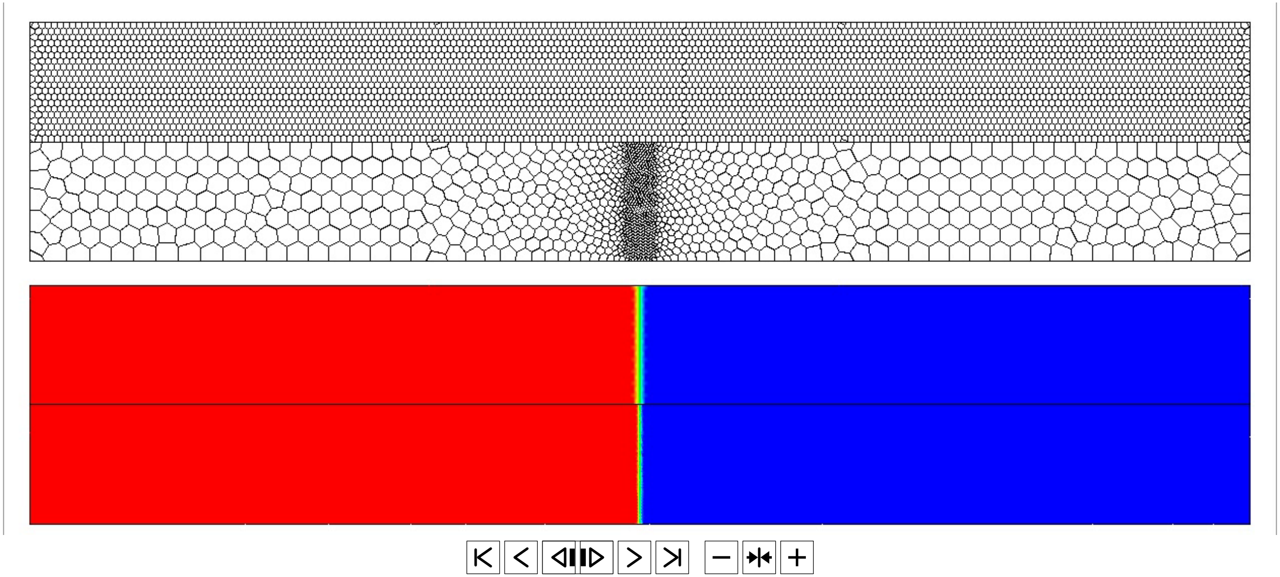
Test problems

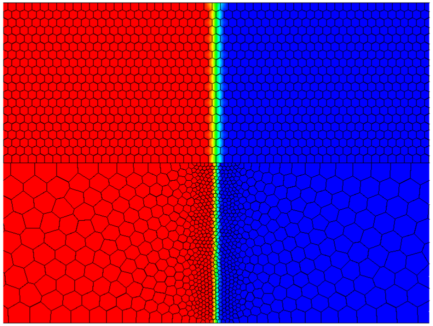
- Sod's shock tube
- Sedov blastwave
- Single material triple point problem

Adaptive ReALE

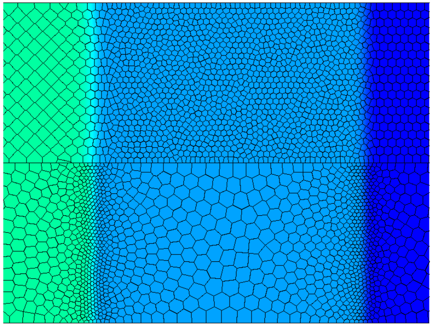
- The parameters of monitor functions depend on test problems
- Mesh convergence criterion in smoothing $|\mathcal{E}^{n+1} - \mathcal{E}^{n}|/\mathcal{E}^{n} < 10^{-4}$
- Comparison with ReALE: the number of generators of ReALE = highest resolution of adaptive ReALE
- Second order conservative remap

Sod's shock tube - mesh and density





$t = 0$



$t = 0.2$ contact and shock

Sod's shock tube - convergence

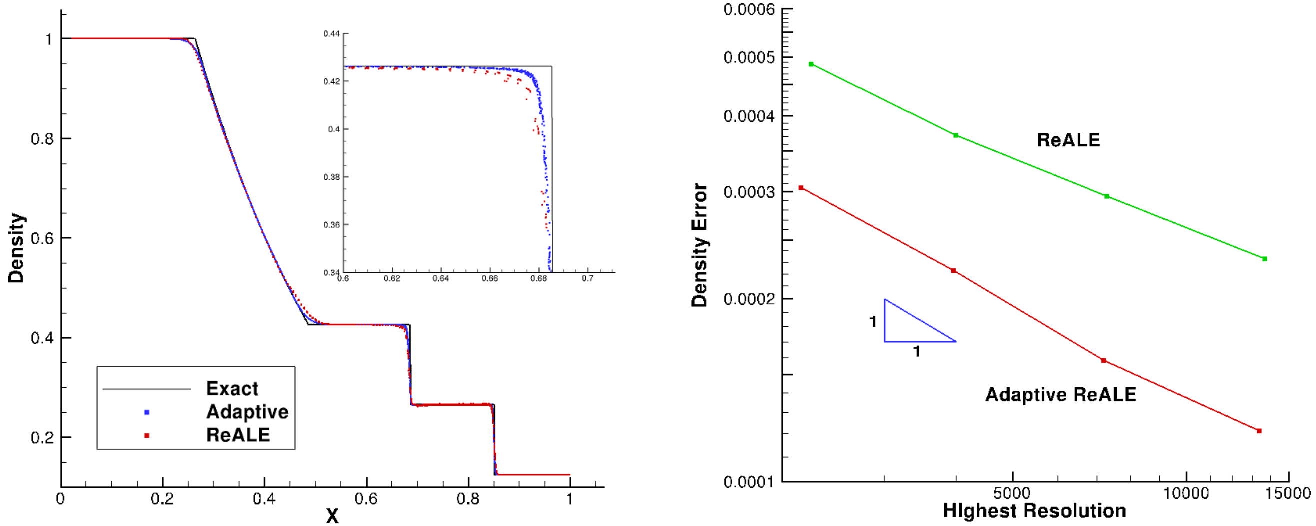
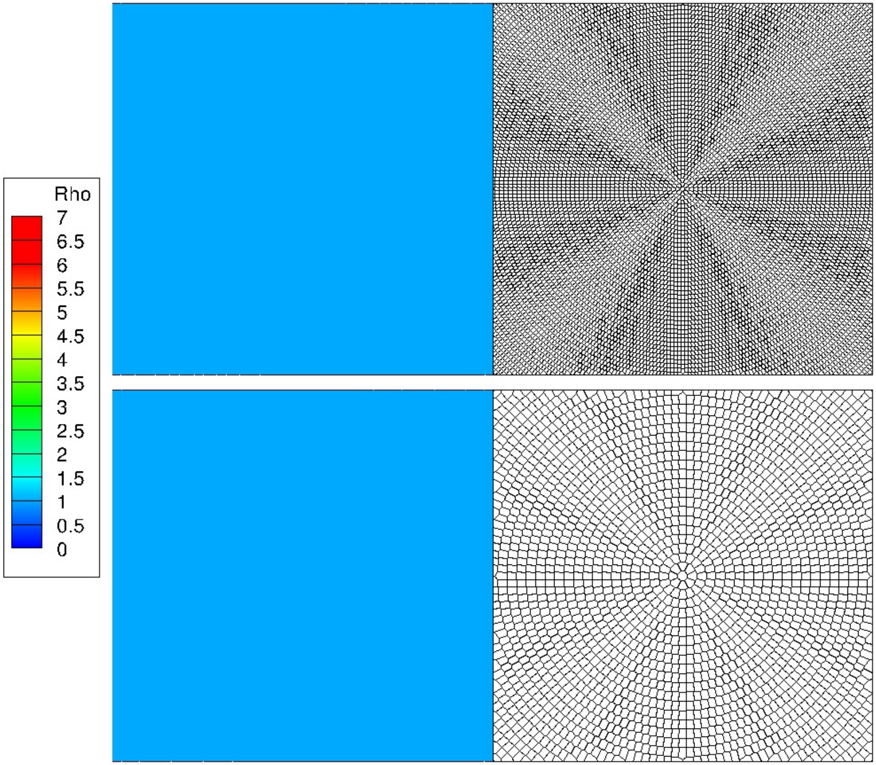


Figure 1 : **Left**: Density, high resolution = 3800 **Right**: L_1 density error

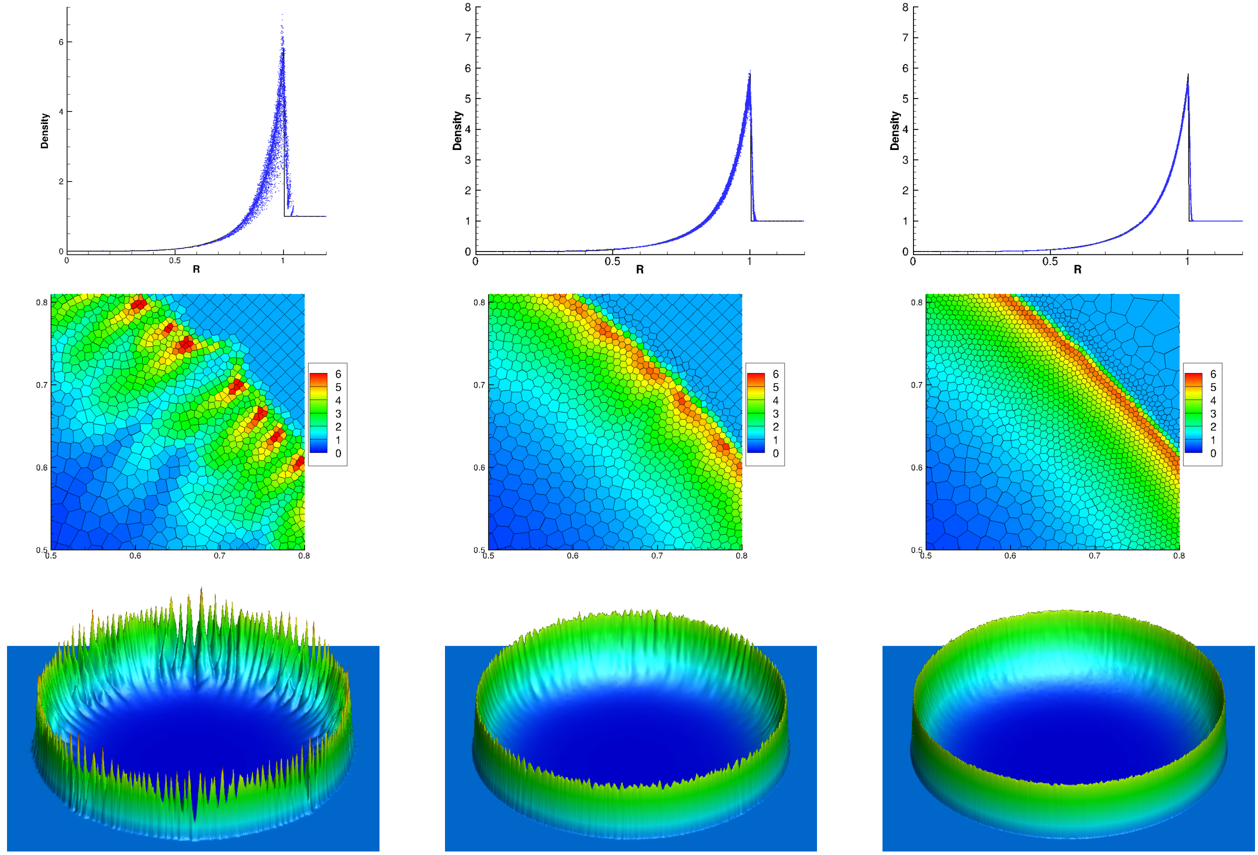
Sedov blastwave - mesh and density

highest resolution = 7260, **Top** ReALE, **Bottom** adaptive ReALE



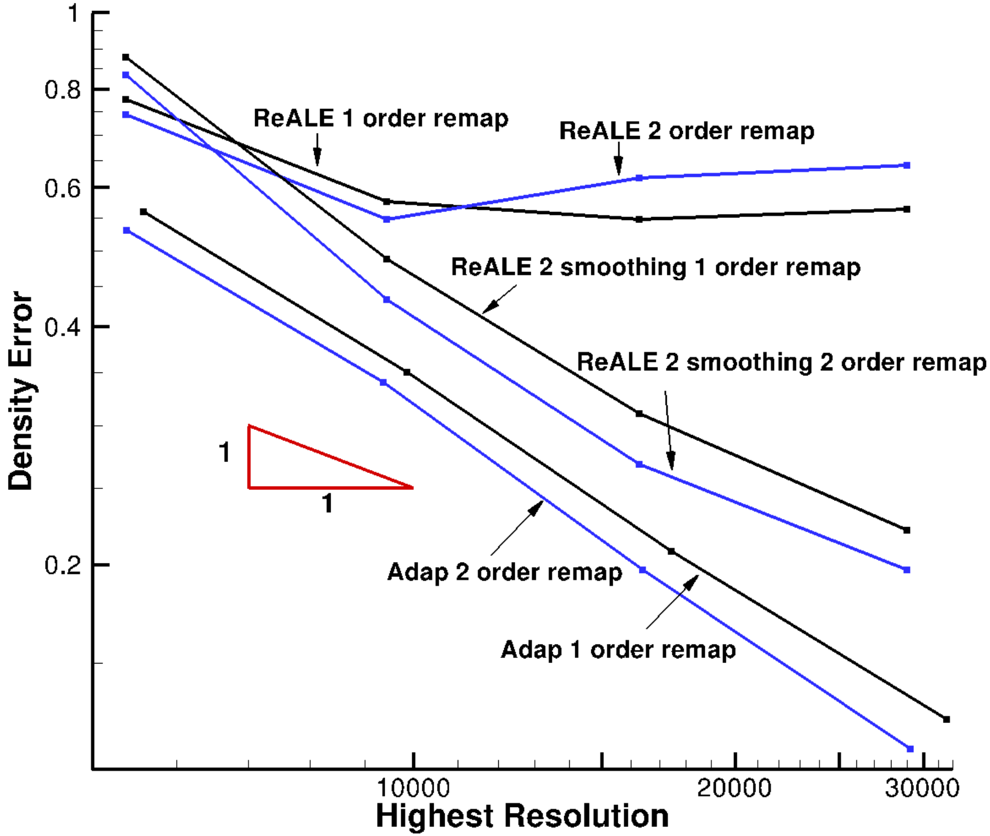
highest resolution = 32580

ReALE | ReALE with 2 smoothing | Adaptive ReALE



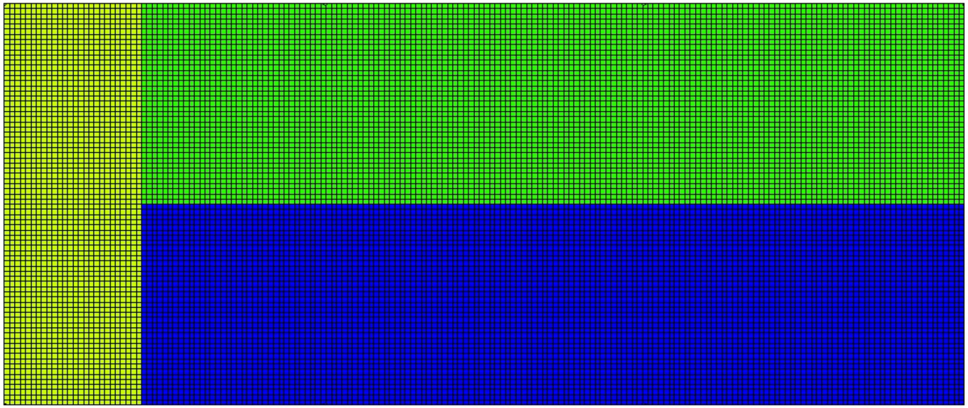
Sedov blastwave - convergence

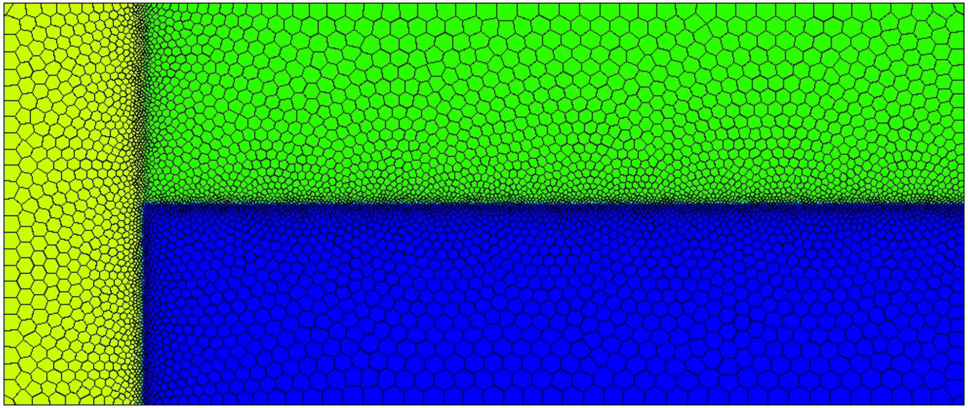
We need to further investigate the convergence of ReALE when the number of generators is large.



Single material triple point problem

$\gamma = 1.4$ for all three regions, highest resolution = 14200, specific internal energy





Single material triple point problem

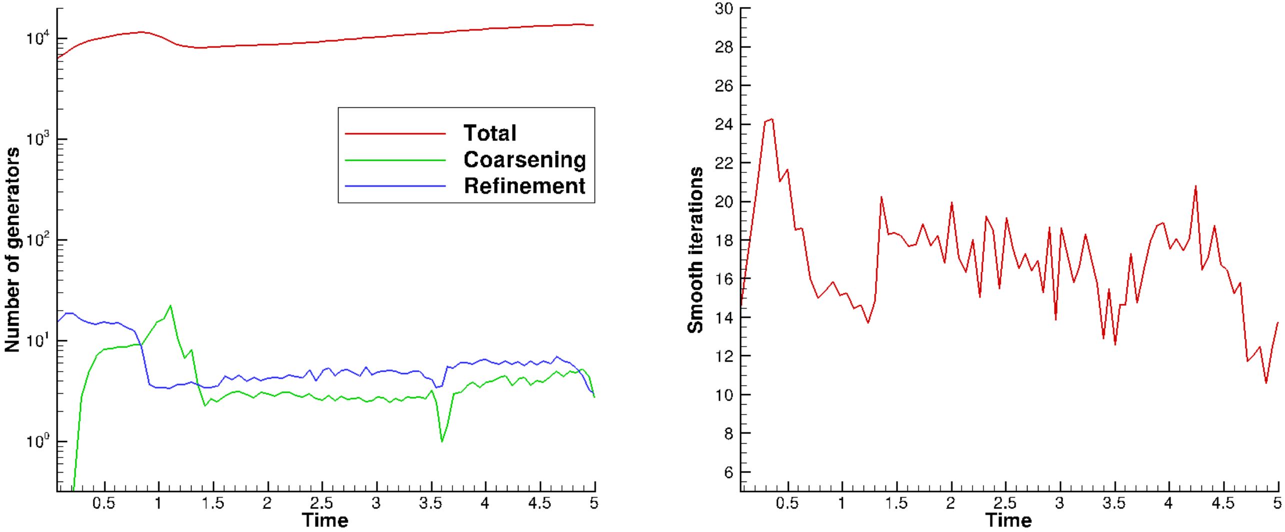


Figure 3 : **Left**: Number of inserted, deleted and total generators **Right**: Number of iterations of global smoothing per step

Conclusions and perspectives

Conclusions

- Insert and delete generators according to a given monitor function
- Local and global smoothing is through dynamic Lloyd-like iterations
- On test problems, adaptive ReALE shows higher accuracy than ReALE

Perspectives

- The convergence of ReALE when the number of generators is large
- Monitor function with less or no parameters
- Performance: In the current implementation, A global Voronoi tessellation is performed after insertion, deletion and smoothing. Local edge swapping may be more efficient in these cases.