ADAPTIVE REFINEMENT

Any adaptive refinement scheme is composed of:

1. Error Indicator/Estimator
   - Interpolation theory
   - Residuals of PDE (adjacent grid)
   - Indicator variable (e.g. Ma, S)
   - Comparison of derivatives (h,p,s)
   - Subgrid: Introduction of further d.o.f. (h,p,s)
2. Aim of refinement
   - Equidistribution of Error
   - Local: $\epsilon^h < c_1 \ \forall \mathbf{x} \in \Omega_{sub}$
   - Refine/Coarsen if: $\epsilon^h > c_r \ , \ \epsilon^h < c_c$
3. Refinement Strategy
   - Mesh Movement (Same Topology)
   - Mesh Enrichment (h,p,s)
   - Mesh Regeneration

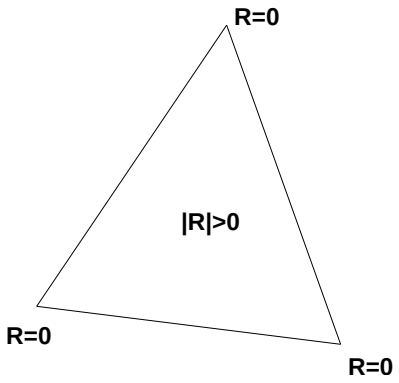
ERROR INDICATORS/ESTIMATORS

Make assumption: $u^h \approx u \Rightarrow$ solution should be 'close'

a) Interpolation theory: for elements of order p

$$\|\epsilon^h\| = \|u - u^h\| = c\ h^{p+1}|u|_{p+1}$$

b) Residuals of PDE (adjacent grid)



c) Element jump in indicator variable (e.g. Ma, S)

d) Comparison of derivatives (h,p,s)

$$u_{,xx} = \frac{1}{h^2}\left(u_{i-1} - 2u_i + u_{i+1}\right) - \frac{1}{12}h^2 u_{,IV}$$

$$u_{,xx} = \frac{1}{12h^2}\left(-u_{i-2} + 16u_{i-1} - 30u_i + 16u_{i+1} - u_{i+2}\right) + \frac{1}{90}h^4 u_{,VI}$$

d) Subgrid: Introduction of further d.o.f. (h,p,s)

....and all work for most cases !

INTERPOLATION THEORY ESTIMATE (1)

H2-seminorm :

$$\|u - u^h\|_0 \leq c \cdot h^2 \cdot |u|_2 \quad ,$$

where

$$|u|_2 = \sqrt{\int_\Omega \sum_{i,j} \left[\frac{\partial^2 u}{\partial x^i \partial x^j}\right]^2 d\Omega} \qquad (*)$$

In 1-D:

$$\epsilon^{RMS} \approx \frac{1}{11} h^2 |u_{,xx}|$$

$\Rightarrow$ Compute second derivatives at nodes
Assume:

$$u = N^k \hat{u}_k \quad ; \quad u_{,ij} = N^k (\hat{u}_{,ij})_k$$

Then use WRM:

$$\int_\Omega N^l N^k (\hat{u}_{,ij})_k d\Omega = \int_\Omega N^l N^k_{,ij} \hat{u}_k d\Omega = -\int_\Omega N^l_{,i} N^k_{,j} \hat{u}_k d\Omega$$

$\Rightarrow$

$$\mathbf{M} \cdot \mathbf{u}_{,ij} = -\mathbf{K} \cdot \mathbf{u}$$

- take max(element nodes) to compute (*)
  ('conservative')

INTERPOLATION THEORY ESTIMATE (2)

For linear elements, constant h, 1-D, at nodes :

$$e_i = h^{-2} \cdot (U_{i+1} - 2 \cdot U_i + U_{i-1})$$

- not dimensionless, i.e. requires user expertise
- strong shocks dominate refinement

$\Rightarrow$ MODIFIED ERROR INDICATOR (1-D)

$$E_i = \frac{|U_{i+1} - 2 \cdot U_i + U_{i-1}|}{|U_{i+1} - U_i| + |U_i - U_{i-1}| + \epsilon \left[|U_{i+1}| + 2 \cdot |U_i| + |U_{i-1}|\right]}$$

- 'eating-up' effect of strong shocks avoided
- dimensionless
- bounded : $0 \leq E_i < 1$ (preset tolerance)
- ϵ : noise/physics filter (hydro-solver dependent)

MODIFIED ERROR INDICATOR (2/3-D)

Generalization to multidimensional situations :

$$E^I = \sqrt{\frac{\sum_{k,l} \left(\int_\Omega N^I_{,k} N^J_{,l} d\Omega \cdot U_J\right)^2}{\sum_{k,l} \left(\int_\Omega |N^I_{,k}| \left[|N^J_{,l} U_J| + \epsilon \left(|N^J_{,l}||U_J|\right)\right] d\Omega\right)^2}}$$

- $1 \leq k, l \leq$ NDIMN

- good for 2-D

- for 3-D unreliable; reason:
  - large local variations in element size
  - large local variations in element shape
  - large local variations in elements surrounding a point

$\Rightarrow$ **IMPROVED ERROR INDICATOR**

$$E^I = \sqrt{\frac{\sum_{k,l} \left(\int_\Omega N^I_{,k} N^J_{,l} d\Omega \cdot U_J\right)^2}{\sum_{k,l} \left(\int_\Omega |N^I_{,k}||N^J_{,l} U_J| d\Omega\right)^2 + \epsilon M_I h_I^{-2} |U_I|}}$$

- M_I: lumped mass-matrix at point I

- h_I: average element length at point I

- Very good performance due to averaging (M_I, h_I)

DETERMINATION OF THE ELEMENT SIZES

Define:

$$D_i^0 = c_n \left(|U_{i+1}| + 2 \cdot |U_i| + |U_{i-1}| \right)$$

$$D_i^1 = |U_{i+1} - U_i| + |U_i - U_{i-1}|$$

$$D_i^2 = |U_{i+1} - 2 \cdot U_i + U_{i-1}|$$

Error on the old grid E^{old}:

$$E_i^{old} = \frac{D_i^2}{D_i^1 + D_i^0}$$

Reduce current element size h^{old} by a fraction ξ to

$$h^{new} = \xi \cdot h^{old}$$

$\Rightarrow$

$$E_i^{new} = \frac{D_i^2 \xi^2}{D_i^1 \xi + D_i^0}$$

$\Rightarrow$

$$\xi = \frac{E^{new}}{E^{old}} \frac{1}{2} \left[\frac{D_i^1 + \sqrt{\left(D_i^1\right)^2 + 4 D_i^0 \frac{E^{old}}{E^{new}} \left[D_i^1 + D_i^0\right]}}{\left[D_i^1 + D_i^0\right]} \right]$$

Remarks:

- Solution smooth $(D^1 \ll D^0) \quad \Rightarrow \quad \xi = \sqrt{\frac{E^{new}}{E^{old}}}$

- Near discontinuity $(D^1 \gg D^0) \quad \Rightarrow \quad \xi = \frac{E^{new}}{E^{old}}$

- Use target E^{targ} as E^{new}

GENERALIZATION TO 2/3-D

$$(D^0)^I_{kl} = h^2 c_n \int_\Omega |N^I_{,k}||N^J_{,l}||U_J| d\Omega \quad ,$$

$$(D^1)^I_{kl} = h^2 \int_\Omega |N^I_{,k}||N^J_{,l} U_J| d\Omega \;\; , \;\; (D^2)^I_{kl} = h^2 | \int_\Omega N^I_{,k} N^J_{,l} d\Omega \, U_J \, |$$

h: 'typical element length'

$\Rightarrow$ Error-matrix $\mathbf{E}$

$$\mathbf{E} = \begin{Bmatrix} E_{xx} & E_{yx} & E_{zx} \\ E_{xy} & E_{yy} & E_{zy} \\ E_{xz} & E_{yz} & E_{zz} \end{Bmatrix} = \mathbf{X} \cdot \begin{Bmatrix} E_{11} & 0 & 0 \\ 0 & E_{22} & 0 \\ 0 & 0 & E_{33} \end{Bmatrix} \cdot \mathbf{X^{-1}}$$

$$\Rightarrow \quad E_{11}, E_{22}, E_{33}, \mathbf{S_1}, \mathbf{S_2} \quad \Rightarrow \quad \xi_1, \xi_2, \xi_3, \mathbf{S_1}, \mathbf{S_2}$$

SMOOTHING OF ELEMENT LENGTHS

Ss.1 For each element *el*: take the average element length h_{el}^{ave} over its nodes:

$$h_{el}^{ave} = \frac{1}{nnoel} \sum_{i=1}^{nnoel} h_i$$

Ss.2 For each point *i*: form the average element length l_i^{ave} over its surrounding elements:

$$l_i^{ave} = \frac{1}{nsuel} \sum_{el=1}^{nsuel} h_{el}^{ave}$$

Ss.3 For each point *i*: obtain the new element length h_i from

$$h_i = min(l_i^{ave}, h_i)$$

$\Rightarrow$ Take a 'point average', not an 'area weighted average'

SMOOTHING OF STRETCHINGS (1)

Stretching direction $\mathbf{s}$ is a vector quantity

Sv.1 For each element el: compute the maximum stetching $\mathbf{s}_{el}^{max}$ encountered at the nodes:

$$\mathbf{s}_{el}^{max} = \max_{i=1,nnoel} \mathbf{s}_i$$

Sv.2 For each element el: form the average element stretching $\mathbf{s}_{el}^{av}$ as

$$\mathbf{s}_{el}^{av} = \frac{1}{nnoel} \sum_{i=1}^{nnoel} \mathbf{s}_i \; sign(\mathbf{s}_{el}^{max} \cdot \mathbf{s}_i)$$

Sv.3 For each point i: compute the maximum stetching $\mathbf{s}_i^{max}$ over its surrounding elements:

$$\mathbf{s}_i^{max} = \max_{el=1,nsuel} \mathbf{s}_{el}^{av}$$

SMOOTHING OF STRETCHINGS (2)

Sv.4 For each point i: form the average element stretching $\mathbf{s}_i$ over its surrounding elements:

$$\mathbf{s}_i = \frac{1}{nsuel} \sum_{el=1}^{nsuel} \mathbf{s}_{el} \; sign(\mathbf{s}_i^{max} \cdot \mathbf{s}_{el})$$

$\Rightarrow$ Take a 'point average', not an 'area weighted average'

INTERPOLATION OF STRETCHING DIRECTIONS

Stretching direction $\mathbf{s}$ is a vector quantity

Si.1 For the element in which the point to be interpolated falls, compute the maximum stetching encountered at the nodes of the element $\mathbf{s}_e$

Si.2 Modify the stretching vectors at the nodes to be

$$\mathbf{s}_n = \mathbf{s}_n sign(\mathbf{s}_e \cdot \mathbf{s}_n)$$

Si.3 Proceed as usual, adding the area-weighted modified stretching vectors

REFINEMENT STRATEGIES

Mesh Movement

- Spring System ($c = c(\epsilon)$)
- Moving Finite Element

Mesh Enrichment

- Uniform
- Uni-directional where possible

Remeshing

H-REFINEMENT vs. REMESHING

	H-Refinement	Remeshing
- Interpolation/Conservation	easy	not easy
- Minimum h-size	easy	not easy
- Directional Refinement	not easy	easy
- Body/Interface Movement	not easy	easy
- Parallelizable	easy	not easy
- Timings $(\mu sec/pt./grid)$	120	1800

H-REFINEMENT

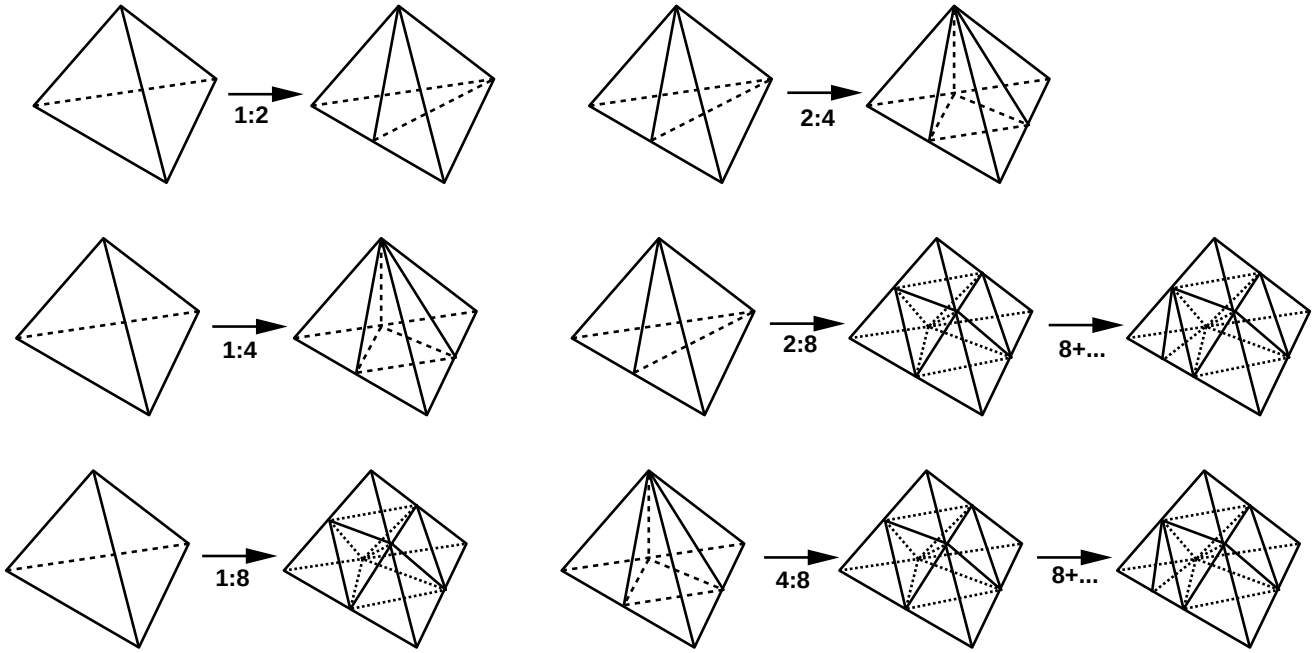


Figure Possible Refinement Cases

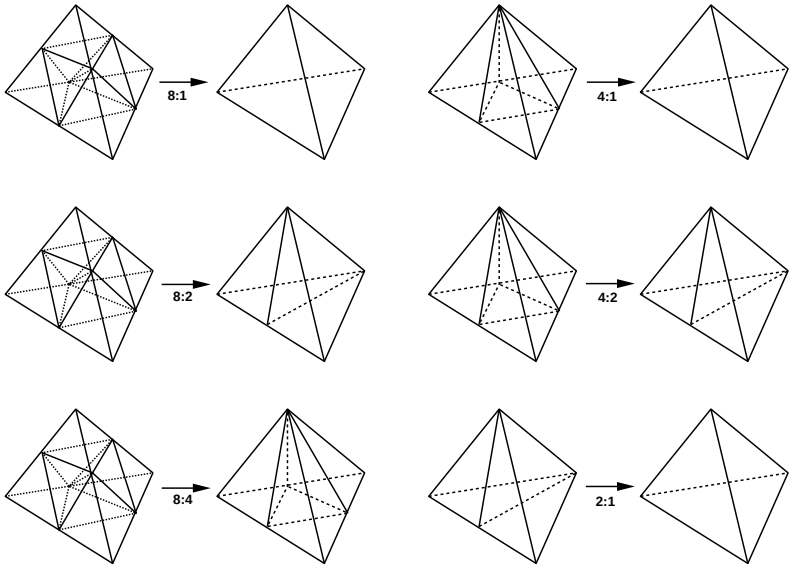


Figure Possible Coarsening Cases

H-REFINEMENT

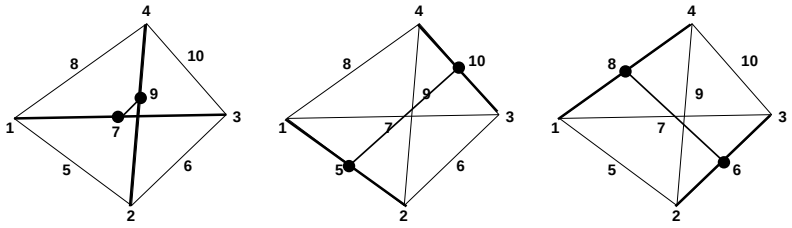


Figure Possible Coarsening Cases

H-REFINEMENT

1. Allow only 1:2, 1:4, 1:8 cases
2. Allow only one level of refinement/coarsening per mesh change

- simplification of logic
- reduction of memory requirements
- increase in speed

GRID LOGIC: INTEGER INFO

Need to be able to reconstruct coarser mesh after refinement

Use 12 integers per element ⇒ `INREME`

`INREME(1:7,IE)`:	son(s) of an element
`INREME(  8,IE)`:	parent of an element
`INREME(  9,IE)`:	position nr. in the parent element
`INREME( 10,IE)`:	element type (parent or son, 1:8, 1:4, 1:2)
`INREME( 11,IE)`:	local refinement level
`INREME( 12,IE)`:	global refinement level

ALGORITHMIC STEPS FOR ONE MESH CHANGE

1. Construction of missing grid information
2. Identification of elements to be refined
3. Identification of elements to be deleted
4. Refinement where needed
5. Deletion where needed

MAIN CPU-INTENSIVE OPERATIONS

- Given tetrahedra, find the sides of the domain
- Find the sides of each tetrahedron
- Determine refinement pattern
- Determine coarsening pattern
- Correct boundary points (CAD-CAM data)
- Renumber the elements

For typical production runs (new mesh every 7 timesteps, 1.5Mtetra), these seemingly trivial operations may account for more than 50% of total running time $\Rightarrow$ optimize

IDENTIFICATION OF ELEMENTS TO BE REFINED

Aim: list of sides where new gridpoints will be introduced

- mark elements for which error exceeds CTORE
- add NLAYR protective layers
- avoid refinement of elements that are too small
- avoid refinement of elements that have been refined too often
- mark sides of elements to be refined
- add sides until an admissible refinement pattern is achieved

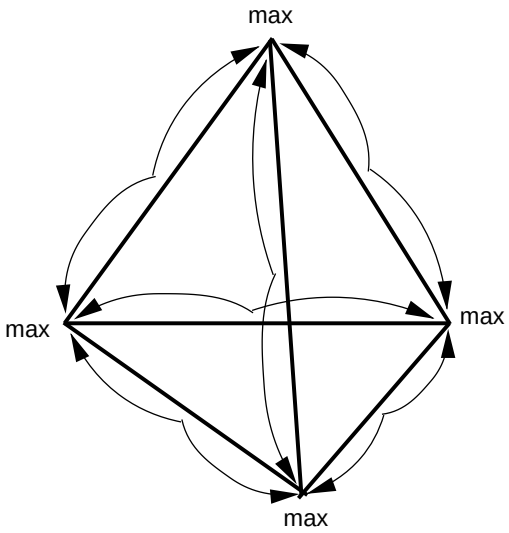
ADDING SIDES FOR ADMISSIBLE REF. (1)

Admissible cases: new points along

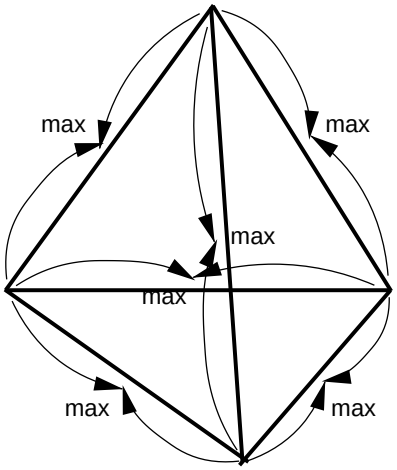
- a side (1:2)
- a face (1:4)
- all sides (1:8)

Element by element algorithm:

- Set the node-array `LNODE(1:4)=0`
- First loop over the sides of the element:
  if the `LSIDE(IS)=1` :
  set `LNODE(IP1)=1`, `LNODE(IP2)=1`
- Second loop over the sides of the element:
  if `LNODE(IP1)=1` and `LNODE(IP2)=1`:
  set `LSIDE(IS)=1`



a) Sides to Points



b) Points to Sides

ADDING SIDES FOR ADMISSIBLE REFINEMENT PATTERN (2)

Practical calculations: up to 15 passes over the mesh required to obtain an admissible set of sides ⇒ expensive

Idea: presort elements

- Add up all the sides marked for refinement in an element
- If 0,1 or 6 sides were marked: do not consider further
- If 4 or 5 sides were marked: mark all sides of this element to be refined
- If 2 or 3 sides were marked: analyze in depth as described above.

⇒ 80%-90% reduction in CPU cost

IDENTIFICATION OF ELEMENTS TO BE DELETED

- mark elements for which error lies below CTODE
- consider only 'parent'-elements to be deleted
- of these, keep only those 'parent'-elements for which all 'son'-elements are to be deleted
- obtain a preliminary list of points to be deleted
- delete points from this list to obtain an admissible coarsening pattern

REFINEMENT OF THE GRID WHERE NEEDED

Given: sides to be refined

- introduce new points along sides to be refined
- interpolate linearly
- introduce the new boundary conditions where needed
- order elements to be refined into groups:
  (1 : 2 , 1 : 4 , 1 : 8 , 2 : 4 , 2 : 8 , 4 : 8)
- refine each group in turn
- correct boundary points using CAD-CAM data

COARSENING OF THE GRID WHERE NEEDED

Given: points to be deleted

- renumber points / boundary conditions to fill up voids in arrays
- order elements to be coarsened into groups:
  (8 : 1 , 8 : 2 , 8 : 4 , 4 : 1 , 4 : 2 , 2 : 1)
- coarsen each group in turn
- renumber elements to fill up voids in arrays
- restructure INREME

ADAPTIVE REMESHING

R.1 Obtain the error indicator matrix

R.2 Compute new element size, stretching and str. direction

R.3 Using old grid as background grid, remesh using the advancing front technique

R.4 If global h-refinement desired: refine further

R.5 Interpolate the solution from the old grid to the new one

GLOBAL REMESHING AND GLOBAL H-REFINEMENT

Idea: Combine the strengths of the two approaches

Remeshing:
- Directional Refinement
- Moving Boundaries

H-Refinement:
- Speed

Even with 1 level of H-Refinement:
Speed-up of 3 (2D), 7 (3D) (!)

GLOBAL H-REFINEMENT

H.1 Construct the missing grid information:
- Points corresponding to sides,
- Sides corresponding to each element,
- Sides corresponding to each boundary face.

H.2 Introduce new points along the sides.

H.3 Form the new elements by subdivision of the old elements into eight.

H.4 Obtain the new boundary conditions from the boundary face/side information.

H.5 Form the new boundary faces by subdivision of the old boundary faces into four.

H.6 Correct the location of the boundary points lying along curved lines/surfaces.

LOCAL REMESHING

Observation: Frequency of distorted elements high
⇒ Global remeshing unnecessary
⇒ Use Local remeshing

L.1 Identify distorted elements in moving layers ⇒ `LEREM(1:NEREM)`.

L.2 Enlarge this region of elements by one layer ⇒ `LEREM(1:NEREM)`.

L.3 Form 'holes' in the present mesh:

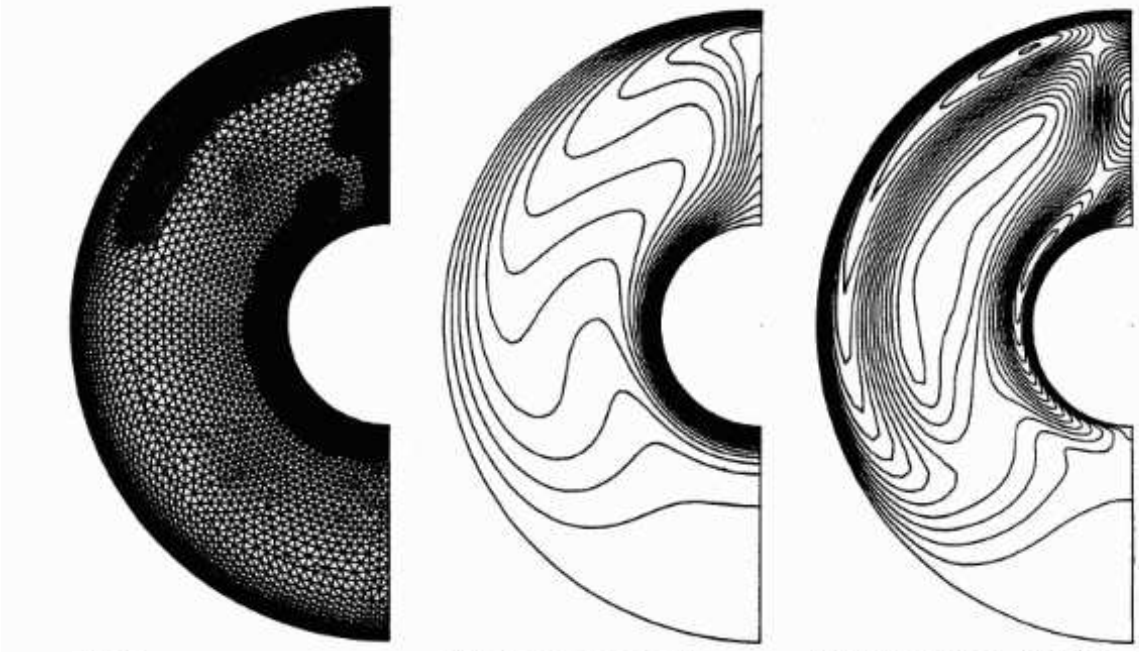
- L.3.1 Form a new background mesh with the elements stored in `LEREM` .
- L.3.2 Delete elements stored in `LEREM` from current mesh.
- L.3.3 Remove all unused points.

L.4 Compute the error indicators and new element distribution.

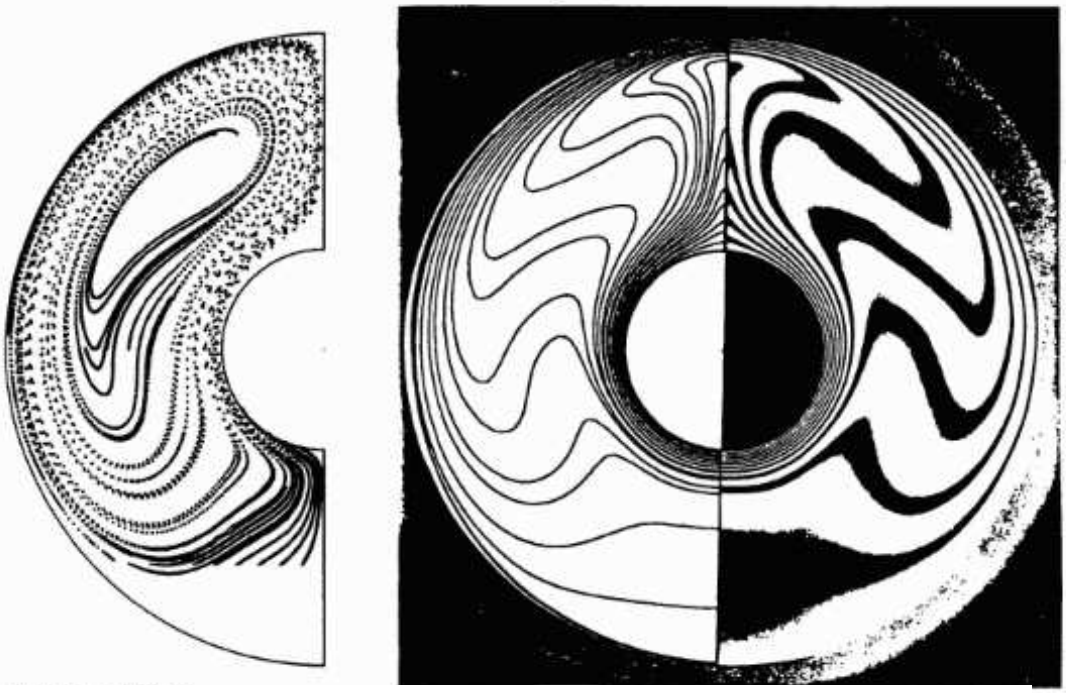
L.5 Regrid 'holes' using advancing front method.

Overall reduction in CPU: 60% (!)

Heated Cylinder

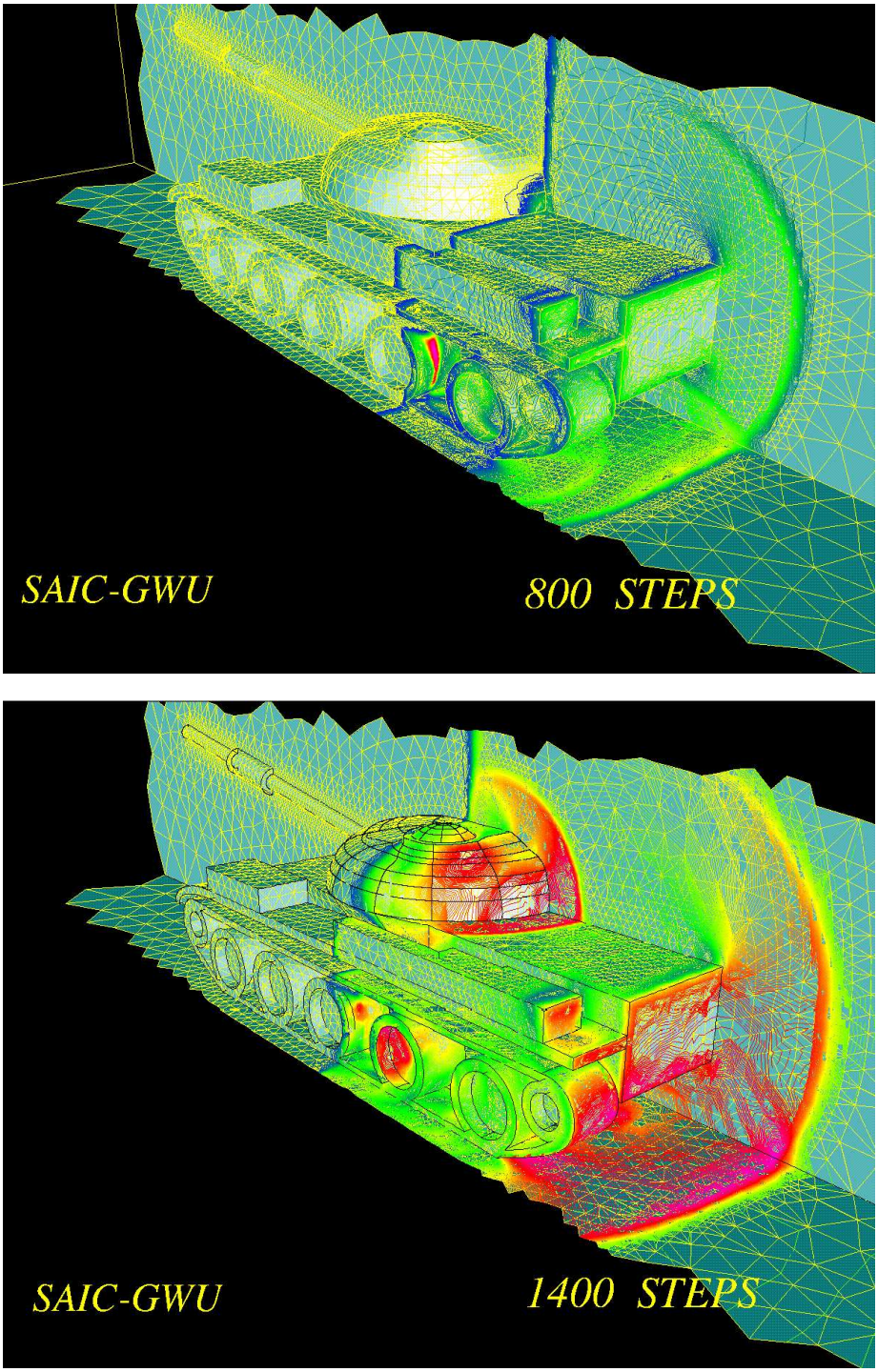


Mesh Temp: dt=0.005 abs(v): dv=1.



Particle Traces Comparison to Experiment

T62



Missile Ejection