Imperial College
London

Mesh Adaptivity and Fluidity

James Percival

AMCG,
Department of Earth Science and Engineering
Imperial College London

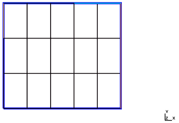
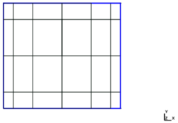
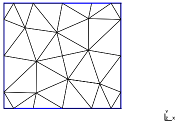
Fluidity Training 2014

Outline

Grids & Meshes

Aside from spectral/particle based methods, computational grids and meshes are ubiquitous in computational science:

- Grids of points for finite difference methods
- Tesselations of subvolumes for finite volume methods
- Tesselations of elements for finite element methods

Grids & Meshes

Meshes/grids:

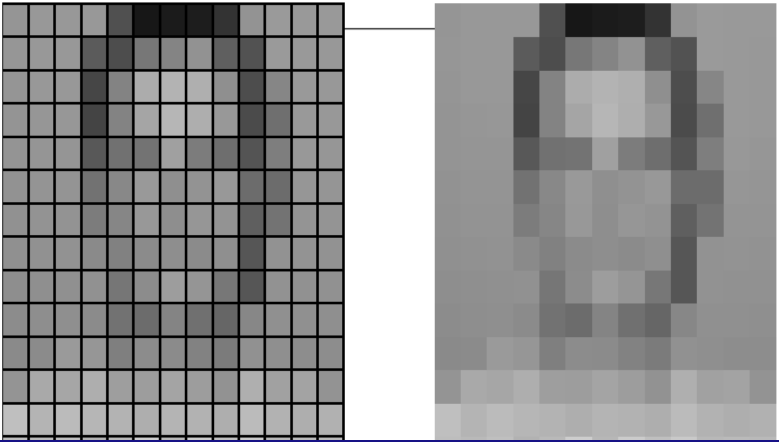
- can be structured or unstructured.
- describe geometry of problem.
- define a local resolvable resolution.
- help determine bounds on solution error.

Frequently fine resolution is only required in limited regions, which change with time.

Idea: progressively modify the mesh to put resolution where it is needed.

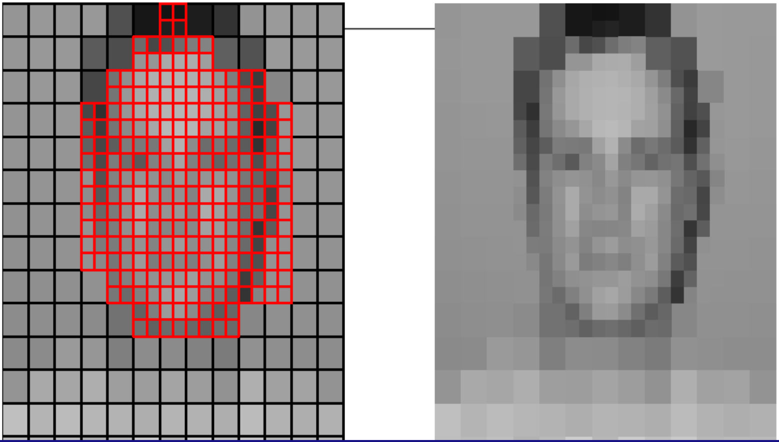
Adaptive Mesh Refinement (AMR)

When this idea is followed through on structured meshes, it is often called adaptive mesh refinement. Blocks are subdivided in areas of refinement. Good for computer memory access, less good for physics.



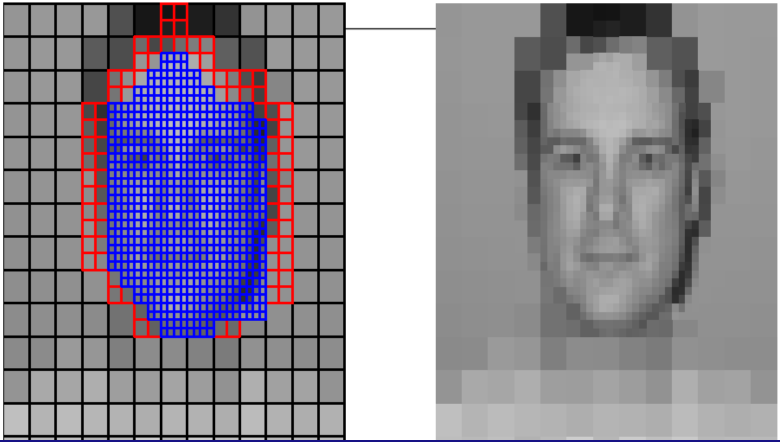
Adaptive Mesh Refinement (AMR)

When this idea is followed through on structured meshes, it is often called adaptive mesh refinement. Blocks are subdivided in areas of refinement. Good for computer memory access, less good for physics.



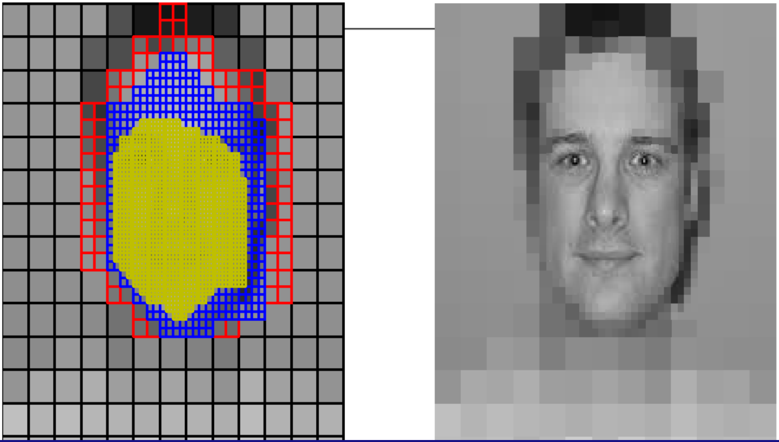
Adaptive Mesh Refinement (AMR)

When this idea is followed through on structured meshes, it is often called adaptive mesh refinement. Blocks are subdivided in areas of refinement. Good for computer memory access, less good for physics.



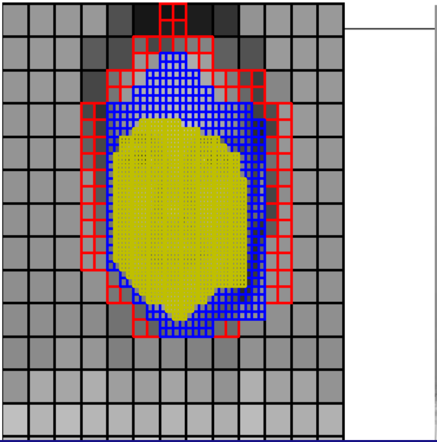
Adaptive Mesh Refinement (AMR)

When this idea is followed through on structured meshes, it is often called adaptive mesh refinement. Blocks are subdivided in areas of refinement. Good for computer memory access, less good for physics.

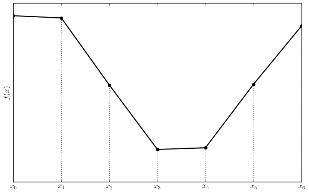


Adaptive Mesh Refinement (AMR)

When this idea is followed through on structured meshes, it is often called adaptive mesh refinement. Blocks are subdivided in areas of refinement. Good for computer memory access, less good for physics.

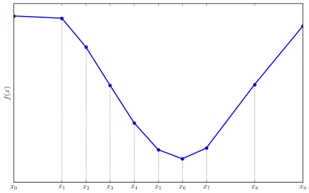


Finite elements: *h* adaptivity



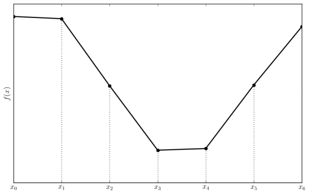
$\Rightarrow$

h adapt



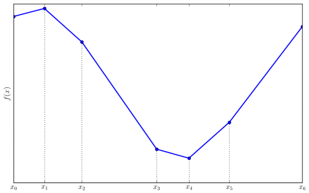
- Adapt by using more, smaller elements (h = step size)
- Analagous to AMR.
- Decreases error estimates, increases complexity (refinement)
- Increases error estimates, decreases complexity (coarsening)
- Implemented in Fluidity.

Finite elements: r adaptivity



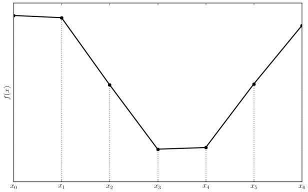
$\Rightarrow$

r adapt



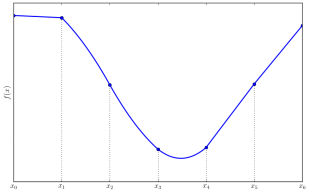
- Adapt by repositioning nodes so that elements are “better” shape
- Decreases error, maintains complexity.
- Implemented in Fluidity.

Finite elements: p adaptivity



$\Rightarrow$

p adapt



- Adapt by using higher/lower order elements as required in different parts of mesh
- Decreases error estimates, increases complexity (refinement)
- Increases error estimates, decreases complexity (coarsening)
- Complex coupling with h adaptivity.
- Not currently implemented in Fluidity

Revision of Céa's lemma

Céa's lemma provides inequality.

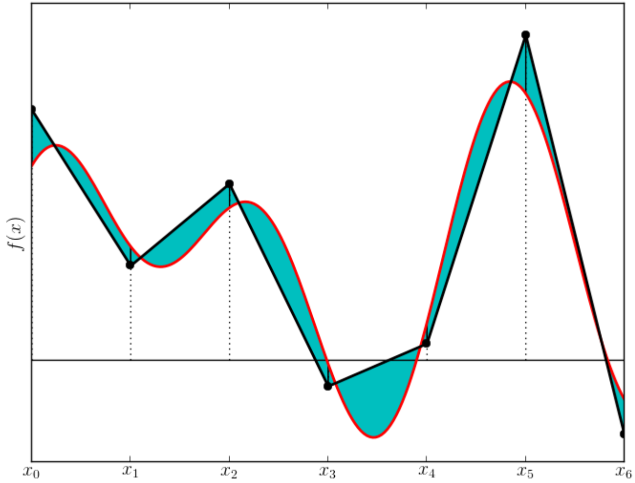
$$\text{error} := \left\|\psi - \psi^{\delta}\right\| \leq \frac{\alpha C}{c} \sup_{\Omega_i} h_i \sup_{x\in\Omega} \left|\frac{\partial^2\psi}{\partial x^2}\right|$$

Used this to show convergence as $h \to 0$. But also a connection to adaptive meshing. Go element by element

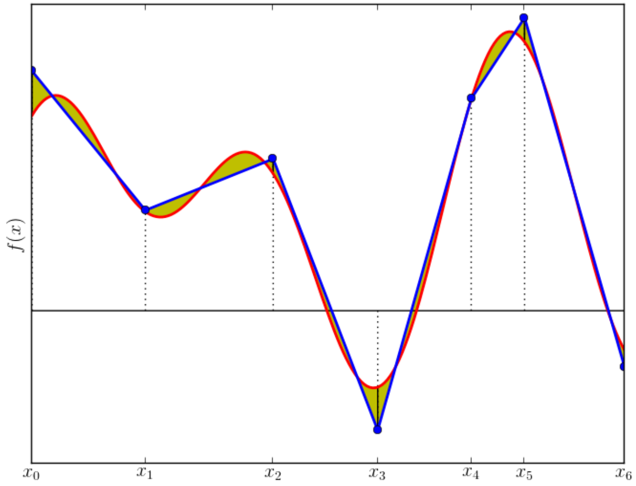
$$\left\|\psi - \psi^{\delta}\right\| \leq \frac{\alpha}{c} \sum_i h_i^2 \sup_{x\in\Omega_i} \left|\frac{\partial^2\psi}{\partial x^2}\right|.$$

One definition of "good" mesh will minimise this representation error estimate. When $\left|\frac{\partial^2\psi}{\partial x^2}\right|$ is small, element size h_i can be large, and vice versa.

Connection to mesh adaptivity



Connection to mesh adaptivity



Connection to mesh adaptivity

Problem:

We don't know true solution, ψ, so can't find $\frac{\partial^2\psi}{\partial x^2}$ etc.

Solution:

We have an existing value of FE solution, ψ^δ, Esimate derivatives from that.

In 2D/3D $\frac{\partial^2\psi}{\partial x^2}$ gets replaced in the formula with the Hessian matrix,

$$\mathcal{H}(\psi) = \begin{pmatrix} \frac{\partial^2\psi}{\partial x^2} & \frac{\partial^2\psi}{\partial x\partial y} & \frac{\partial^2\psi}{\partial x\partial z} \\ \frac{\partial^2\psi}{\partial x\partial y} & \frac{\partial^2\psi}{\partial y^2} & \frac{\partial^2\psi}{\partial y\partial z} \\ \frac{\partial^2\psi}{\partial x\partial z} & \frac{\partial^2\psi}{\partial y\partial z} & \frac{\partial^2\psi}{\partial z^2} \end{pmatrix}.$$

Mesh optimization

Taking useful approximations to the error estimate, (see Chapter 7 of Fluidity manual) our goal for a mesh with “nicely spread” error estimate is

$$\frac{1}{\epsilon}\boldsymbol{v}_k^T\mathcal{M}\boldsymbol{v}_k = 1$$

where the $\boldsymbol{v}$s are the (vector) edges of the mesh, $\mathcal{M} = \mathcal{M}\left(\psi^\delta\right)$ is a function of the Hessian matrix (called the mesh metric tensor). The interpolation error bound, ϵ, is a normalizing factor which also nondimensionalizes the problem.

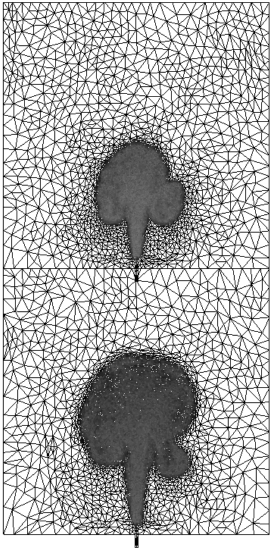
Notes on the interpolation error bound

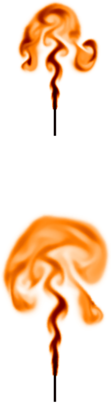
The ϵ has the same physical units as the field, ψ, so can sensibly be specified in either:

1. absolute units (eg. 1 m/s for a velocity field) or,
2. relative units (eg. 5% of the local value of ψ^{δ}).

It is also the principle control on the *number* of nodes and elements required. It is thus often hard to guess without experimentation, and simulation data from fixed resolution runs. In the absence of other information, a fraction (e.g. 10%) of the total variation in ψ expected across the entire domain may be a suitable first choice.

Examples of the Results





Points to remember

- Mathematics has been given for a single variable, but in practice CFD involves multiple prognostic variables.
- Extension trivial, but each field requires own interpolation error bound.
- Haven't discussed the choice function in $\mathcal{M}$.
- Implementation in parallel introduces additional complications
- See Fluidity manual for more details.
- Can minimize error estimate, but this may make numerics tricky. Answer through additional constraints.

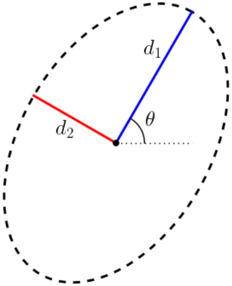
Metric Tensor Bounds

Usually desriable to have some additional constraints on the target lengths in the mesh error metric tensor. Fluidity accepts 2 inputs, giving the equation of bounding ellipses/ellipsoids on the edge lengths in symmetric positive definite tensorial form:

$$\Phi = \begin{pmatrix} a_{11} & a_{21} \\ a_{21} & a_{22} \end{pmatrix} = R \begin{pmatrix} d_1 & 0 \\ 0 & d_2 \end{pmatrix} R^T$$

$$R = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}$$

$$\begin{pmatrix} x & y \end{pmatrix} \Phi^{-1}\Phi^{-1} \begin{pmatrix} x \\ y \end{pmatrix} = 1.$$



Notes on Metric Tensor Bounds

- Bounds are soft limits, not hard ones.
- Interpolation error bounds should be primary control.
- Maximum edge lengths often come from the geometry.
  - Also need to have enough resolution for phenomena to form.
- Minimum edge lengths often come from the physics.
  - Also bad idea to have too many lengthscales in problem at once.
- Example in domain with sides $L \times L \times H$

$$\text{max.} = \begin{pmatrix} \frac{L}{10} & 0 & 0 \\ 0 & \frac{L}{10} & 0 \\ 0 & 0 & \frac{H}{10} \end{pmatrix}, \quad \text{min.} = \begin{pmatrix} \frac{L}{100} & 0 & 0 \\ 0 & \frac{L}{100} & 0 \\ 0 & 0 & \frac{H}{100} \end{pmatrix}$$

Metric Advection

The new mesh is optimized to have a small error metric at time it is calculated. However, remeshing each time step would be both expensive and potentially damaging to the solution. Ideally would like to have bound on the error estimate which remains reasonable as the simulation progresses.
For advection dominated problems we can do this by treating the metric tensor as "just another variable"

$$\frac{\partial \mathcal{M}}{\partial t} + \boldsymbol{u}\left(t_0\right) \cdot \nabla \mathcal{M} = 0, \quad t \in [t_0, t_f]$$

And attempt to set

$$\frac{1}{\epsilon} \max_{\tau \in [t_0, t_f]} \left(\boldsymbol{v}_k^T \mathcal{M}\left(\tau\right) \boldsymbol{v}_k \right) = 1.$$

Notes on Metric Advection

- Metric advection increases limit on the period between adapts (usually)
  - Rule of thumb for transient problem: 10 - 20 time steps.
- Avoids issues with dissipation & balance linked to frequent mesh adaptation.
- Fairly robust to choice of advection scheme, so long as it's stable
  - (see next set of talks)
- May increase problem sizes (larger areas with high resolution)
- May misbehave if the physics of the system bifurcates.

Metric Gradation

- Often a bad idea to have length scales changing very rapidly between neighbouring elements. (conditioning)
- Post-processing of $\mathcal{M}$ can enforce smooth increases in the metric away from minima:
  - Isotropic $\|\boldsymbol{v}_i\|/\|\boldsymbol{v}_j\| \leq 1.5$ if i, j are neighbouring edges (default)
  - anisotropic $\boldsymbol{v}_i \cdot \underline{\Gamma} \cdot \boldsymbol{v}_j \leq \|\boldsymbol{v}_j\|^2$ if i, j are neighbouring edges

Notes on Metric Gradation

A typical choice would be

$$\underline{\Gamma} = \begin{pmatrix} 1.5 & 0 \\ 0 & 1.5 \end{pmatrix} \text{ or, to same result } \begin{pmatrix} 1/1.5 & 0 \\ 0 & 1/1.5 \end{pmatrix}$$

- The default gradation is good unless there's a special reason to have a very smooth variation.
- Anisotropic gradation is more suited to anistropic problems (fronts/filaments/high aspect ratios)
- Sometimes want to limit individual element aspect ratios as well.

Mesh to Mesh interpolation

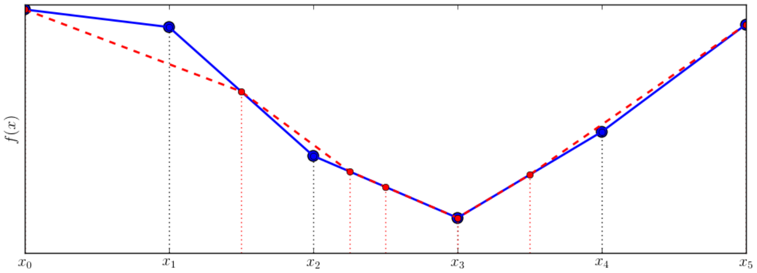
Having found an improved mesh to represent our solutions, must still transfer variable data (velocities, pressures etc) from the old mesh onto the new one. This is the mesh to mesh interpolation problem. Ideal method would:

- be fast computationally
- be conservative where relevant
- remain bounded (for data which have meaningful bounds)
- return the original data for the original mesh

Consistent Interpolation

Conceptually simplest scheme is to set nodes of the new mesh to the function values on the old mesh,

$$\psi_i^{\text{new}} = \psi^{\text{new}}(\boldsymbol{p}_i) = \sum_j N_j^{\text{old}}(\boldsymbol{p}_j)\,\psi_j^{\text{old}}$$



Notes on Consistent Interpolation

- Fast.
- Nonconservative.
- Bounded.
- Preserves data under identity mesh transformation.
- Doesn't like discontinuous data.

Galerkin projection

A second option is to treat the problem

$$\psi^{\mathsf{new}} = \psi^{\mathsf{old}}$$

as a standard finite element problem. Obtain Galerkin projection equation

$$\sum_j \int_\Omega N_i^{\mathsf{new}} N_j^{\mathsf{new}} \psi_j^{\mathsf{new}} dV = \sum_k \int_\Omega N_i^{\mathsf{new}} N_k^{\mathsf{old}} \psi_k^{\mathsf{old}} dV$$

Left hand side is standard sparse mass matrix. The right hand side is more complicated, but can be dealt with by supermeshing.

Notes on Galerkin projection

- Significantly slower than consistent interpolation.
- Conservative.
- Not bounded (especially near discontinuities/extrema)
- Preserves data under identity mesh transformation.

Summary

- Fluidity mesh adaptivity places resolution in regions of high curvature
- Minimizes the interpolation error estimate for a given number of nodes
- Uses *hr* adaptive scheme.
- Interpolate data from old to new mesh using chosen scheme.
- For a new problem, usually best to do fixed resolution runs first.

Imperial College
London

References

- Fluidity Manual
  AMCG
  2014
- C. C. Pain, A. P. Umpleby, et al. Tetrahedral mesh optimisation and adaptivity for steady-state and transient finite element calculations. *Computer Methods in Applied Mechanics and Engineering*, 2001.
- P. E. Farrell and J. R. Maddison. Conservative interpolation between volume meshes by local Galerkin projection. *Computer Methods in Applied Mechanics and Engineering*, 2011.