AMR for Fluids and Other Applications

Stefano Fochesatto

University of Alaska Fairbanks

April 17, 2025

Overview

Definition (Linear Variational Problem)

Find $u \in H^1_{g_D}$ such that,

$$a(u, v) = F(v) \text{ for all } v \in H^1_0.$$

Where $a(\cdot, \cdot) : H^1(\Omega) \times H^1(\Omega) \to \mathbb{R}$ is a bilinear form, and $F(\cdot) : H^1(\Omega) \to \mathbb{R}$ is a bounded linear form.

Theorem (Cea's Lemma; Ex: Elman et al. 2005)

Let u be the solution to a linear variational problem on H^1 and u_h be the finite element solution on S^h. If a is continuous and coercive then there exists constants $\gamma, \alpha > 0$ such that,

$$\|u - u_h\|_{H^1} \leq \frac{\gamma}{\alpha} \min_{v \in S^h} \|u - v\|_{H^1} .$$

Let $\pi_h(u)$ be the interpolant of u in S^h then,

$$\|u - u_h\|_{H^1} \leq \frac{\gamma}{\alpha} \|u - \pi_h(u)\|_{H^1} .$$

Theorem (1; Ex: Elman et al. 2005)

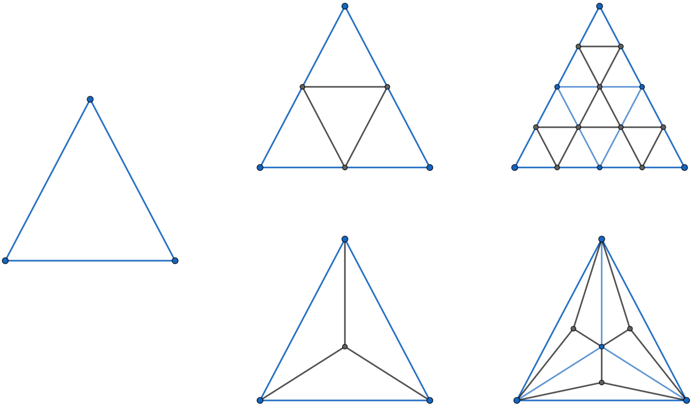
Let T_h be a triangulation and h_k be the largest length and θ_k be the minimum angle of $\triangle_k \in T_h$, then there exists some constant C_2

$$\|\nabla(u - \pi_h(u))\|_{L_2}^2 \leq C_2 \sum_{\triangle_k \in T_h} \frac{1}{\sin^2 \theta_k} h_k^2 \left\| D^2 u \right\|_{\triangle_k}^2 .$$

- By estimation of interpolation error and the Bramble-Hilbert Lemma.

Definition (shape regularity; Ex: Elman et al. 2005)

A sequence of triangulations $\{T_h\}$ is shape regular if there exists a minimum angle $\theta_* \neq 0$ such that every element in T_h satisfies $\theta_T \geq \theta_*$.



Shape Regularity

- All together ...

$$
\begin{aligned}
\|u-u_h\|_{H^1} &\leq \frac{\gamma}{\alpha}\|u-\pi_h(u)\|_{H^1} \qquad \text{(Céa's Lemma)},\\
&\leq \frac{\gamma}{\alpha}\sqrt{1+C_1}\,\|\nabla(u-\pi_h(u))\|_{L_2} \qquad \text{(Poincaré-Friedrichs)},\\
&\leq \frac{\gamma}{\alpha}\sqrt{1+C_1}\left(C_2\sum_{\triangle_k\in T_h}\frac{1}{\sin^2\theta_k}h_k^2\left\|D^2u\right\|_{\triangle_k}^2\right)^{\frac{1}{2}} \qquad \text{(Th.1)},\\
&\leq \frac{\gamma}{\alpha}\sqrt{(1+C_1)C_2}\frac{1}{\sin\theta_*}h\left\|D^2u\right\|_{\Omega} \qquad \text{(shape regularity)}.\\
&= O(h).
\end{aligned}
$$

- A different proof shows $O(h^2)$ for L_2.

- A good S^h minimizes the interpolation error.

$$f(x,y) = \sqrt{1-x^2} \quad \text{on} \quad \Omega = [-1,1]^2$$

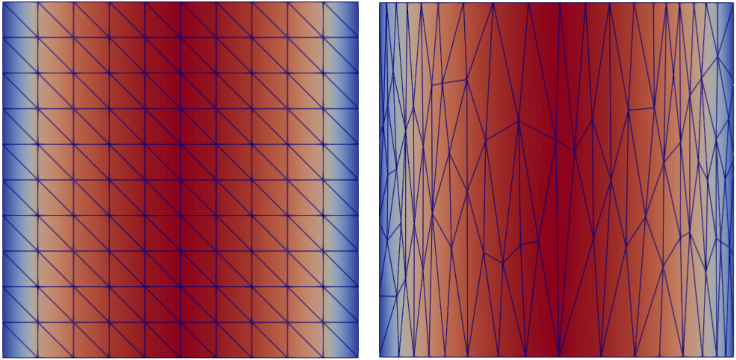


Figure: Interpolation of anisotropic f with roughly the same elements.

- A good refinement scheme should also focus on interpolation error.

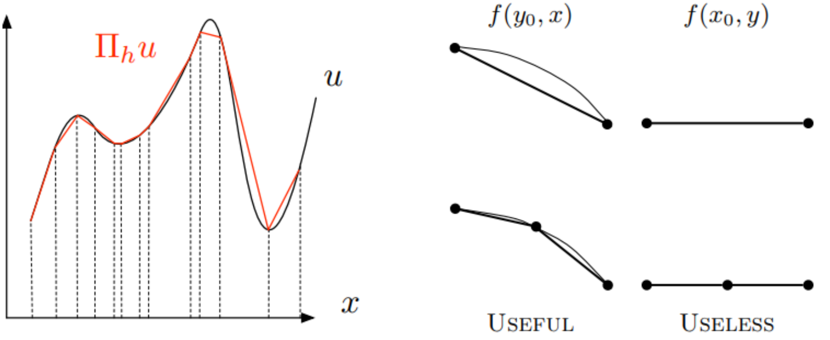


Figure: (Alauzet, 2010)

Tagging Schemes

1. **Solve**: Compute the solution on the current mesh.
2. **Estimate**: Estimate error for each element.
3. **Tag**: Tag elements for refinement/coarsening based on estimate.
4. **Refine**: Refine/coarsen mesh.

- Babuška-Rheinboldt error estimator (for Poisson),

$$\eta_K^2 = h_K^2 \int_K \left| f + \nabla^2 u_h \right|^2 dx + \frac{h_K}{2} \int_{\partial K \setminus \partial \Omega} [\![\nabla u_h \cdot \mathbf{n}]\!]^2 \, ds$$

- Refine and coarsen in a way where error is equidistributed (Bangerth & Rannacher, 2003)

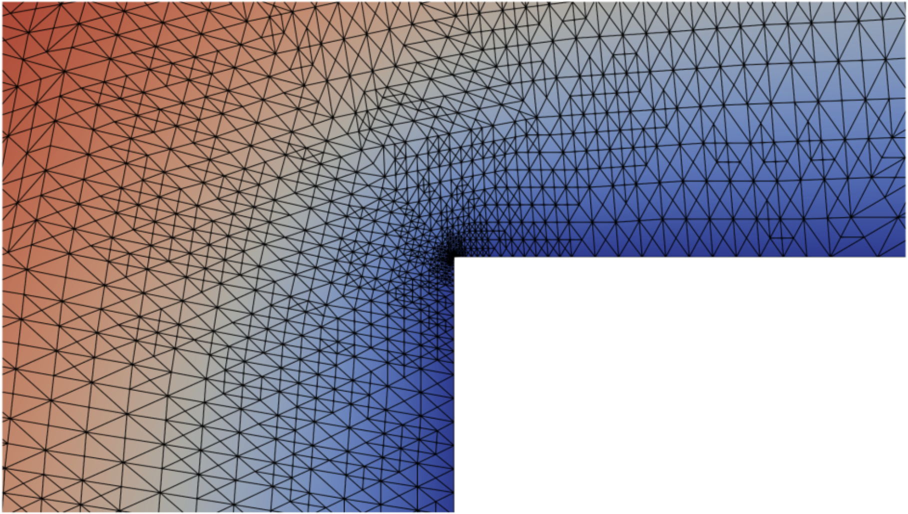


Figure: L-Shaped Homogeneous Dirichlet Poisson Problem.
(Farrell, 2024)

- Mark and refine functionality is implemented in Firedrake via Netgen through ngsPETSc (Zerbinati et al. 2024). 2D and 3D

```
import netgen
mesh = Mesh(ngmesh)
...
AdaptedMesh = mesh.refine_marked_elements(indicator)
```

- The SBR algorithm (Plaza & Carey. 1998) is available in PETSc with bindings in petsc4py (or VIAMR). 2D only

```
import VIAMR
...
AdaptedMesh = VIAMR.refinemarkedelements(mesh, indicator)
```

- Neither implementation can coarsen, so not ideal for time dependent problems.

Metric Based Adaptation

- Let $\Omega \subset \mathbb{R}^n$, and let $\mathbf{M} = \{\mathcal{M}(x)\}_{x\in\Omega}$ be a Riemannian Metric Space, where $\mathcal{M}(x) : \Omega \to \mathbb{R}^{n\times n}$ is an SPD matrix.

- Notions of distance, volume, and angle are derived from **M** and used during mesh generation to drive adaptivity.

$$d_{\mathbf{M}}(a,b) = \int_0^1 \|\gamma'(t)\|_{\mathcal{M}}\,\mathrm{d}t = \int_0^1 \sqrt{ab^T\,\mathcal{M}(a+tab)\,ab}\,\mathrm{d}t.$$

$$|K|_{\mathbf{M}} = \int_K \sqrt{\det\mathcal{M}(x)}\,\mathrm{d}x\,.$$

Geometric Interpretation

$$\mathcal{M}(x)^{-1/2} = U\Lambda^{-1/2}U^{-1}(x) \quad \text{where} \quad \Lambda = I(\lambda_1^{-1/2}, \lambda_2^{-1/2}, \ldots)$$

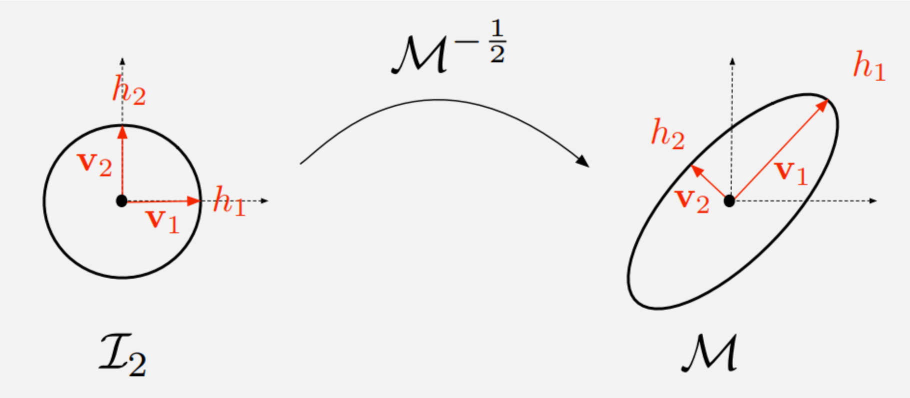


Figure: (Alauzet, 2010)

Geometric Interpretation

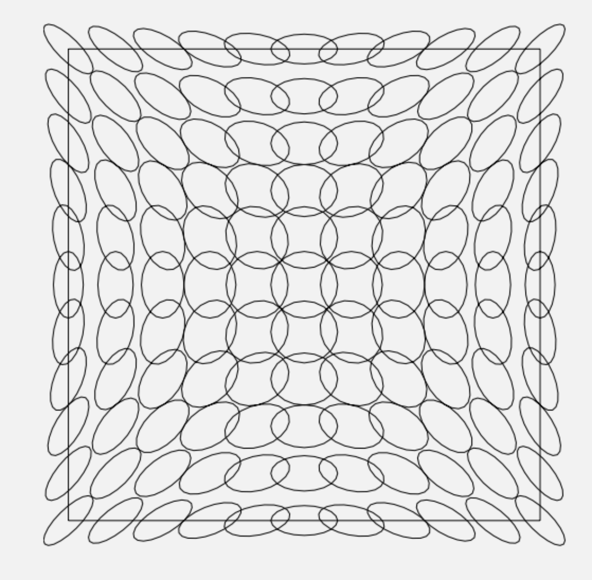


Figure: (Alauzet, 2010)

Geometric Interpretation

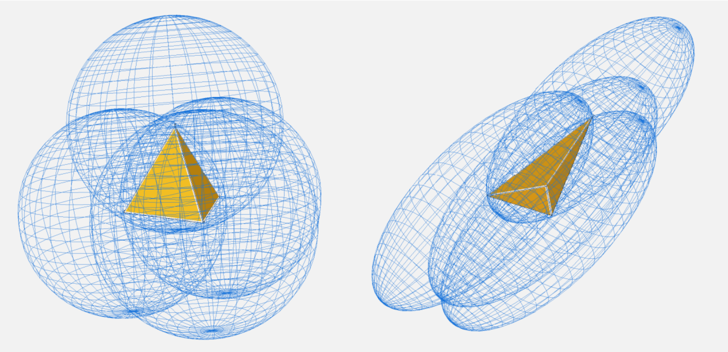


Figure: (Alauzet, 2010)

Isotropic Metrics and Operations

- Isotropic metrics should treat each dimension the same,

$$\mathcal{M}(x) = U(I\lambda(x))U^{-1}.$$

- We can intersect and average metrics to create new metrics.

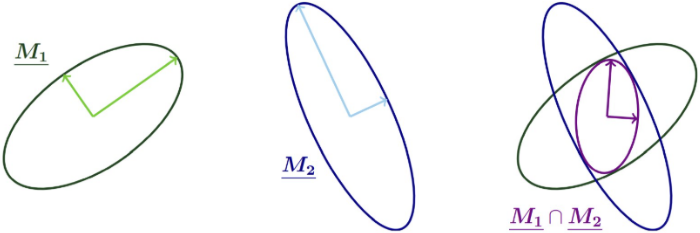


Figure: Metric Intersection (Wallwork, 2021)

Theorem (Alauzet, 2010)

Let u be a twice differentiable function on Ω with $H_u(x)$ its Hessian and let $|H_u(x)| = U\,|\Lambda|\,U^{-1}$. If $\mathcal{H}$ is a unit mesh of Ω generated with respect to the metric,

$$\frac{c_n\,|H_u|}{\epsilon}(x)$$

where c_n depends on dimension of Ω then $\mathcal{H}$ is optimal within ϵ w.r.t controlling linear interpolation error in L^∞ norm.

- Methods exists for interpolating discretely defined metrics and recovering hessians from linear FE solutions.

- Parallel metric based adaptation is implemented in PETSc (Wallwork et al. 2022) and has been ported into Firedrake with the Animate library.

```
import animate
...
P1_ten = TensorFunctionSpace(mesh, "CG", 1)
metric = RiemannianMetric(P1_ten)
metric.set_parameters(metric_params)
metric.compute_hessian(c)
metric.normalise()
adapted_mesh = adapt(mesh, metric)
```

Time Dependent Metric Adaptation

1. Perform hessian based adaptation on the initial time step.
2. Solve the problem on a specified sub-interval.
3. Compute the hessian based metric for each solution in the sub interval.
4. Intersect the metrics and adapt the mesh.
5. Transfer the solution to the new mesh (interp or project)
6. Re-solve the problem on the sub-interval.
7. Repeat 2-6 on the next sub-interval until we reach the end of the simulation.

Demo

Time Dependent Metric Adaptation (Metric Advection)

1. Perform hessian based adaptation on the initial time step.
2. Solve the fluid problem on a specified sub-interval.
3. Solve the advection equation on the initial metric with fluid velocities for the duration of the sub-interval.
4. Intersect the metrics and refine the mesh.
5. Re-solve the problem on the sub-interval.
6. Repeat 2-5 on the next sub-interval until we reach the end of the simulation.

Demo

SUPG Metric Advection

- Solve for metric M where,

$$\frac{\partial m_{ij}}{\partial t} + u \cdot \nabla m_{ij} = 0.$$

- Implicit-Euler weak form,

$$\int_\Omega \left(m^{n+1}\phi + \Delta t (u \cdot \nabla m^{n+1})\phi \right) dx = \int_\Omega m^n \phi \, dx,$$

- SUPG stabilisation for CG finite elements modifies,

$$\phi \to \phi + \tau \, u \cdot \nabla \phi.$$

- The amount of added diffusion is controlled by τ.
- $\tau = \frac{h}{2|u|}$ for pure advection, $\tau = \frac{h|u|}{6K}$ for advection-diffusion.