MATH 676

–

Finite element methods in scientific computing

Wolfgang Bangerth, Texas A&M University

Lecture 30:

The *step-26* tutorial program
–
The heat equation

Part 2: Adaptive meshes for time dependent problems

Spatial discretization

Recall that we needed to solve:

$$\frac{u^n-u^{n-1}}{k_n}-\Delta\left((1-\theta)u^{n-1}+\theta u^n\right) \;=\; (1-\theta)f(x,t_{n-1})+\theta f(x,t_n)$$

or equivalently:

$$u^n-\theta k_n\Delta u^n \;=\; u^{n-1}+(1-\theta)k_n\Delta u^{n-1}+(1-\theta)k_n f(x,t_{n-1})+\theta k_n f(x,t_n)$$

We want to use functions on time dependent meshes:

$$u^n(x) \;\approx\; u_h^n(x) \;:=\; \sum_j U_j^n\varphi_j^n(x) \qquad \in V_h^n$$

$$u^{n-1}(x) \;\approx\; u_h^{n-1}(x) \;:=\; \sum_j U_j^{n-1}\varphi_j^{n-1}(x) \qquad \in V_h^{n-1}$$

How to treat the previous solution?

From the strong form...

$$u^n - \theta k_n \Delta u^n = u^{n-1} + (1-\theta) k_n \Delta u^{n-1} + (1-\theta) k_n f(x, t_{n-1}) + \theta k_n f(x, t_n)$$

...we get the discrete weak form:

$$(\varphi_i^n, u_h^n) + \theta k_n (\nabla \varphi_i^n, \nabla u_h^n) = (\varphi_i^n, u_h^{n-1}) + \ldots \qquad \forall i = 1 \ldots \dim V_h^n$$

Problem:

- $u_h^{n-1}(x) := \sum_j U_j^{n-1} \varphi_j^{n-1}(x) \quad \in V_h^{n-1}$ defined on mesh *n-1*
- Need to integrate functions from different meshes

- In general: very expensive, not practically useful
- Hierarchically related meshes in deal.II: feasible but awkward (see step-28 for an example)

How to treat the previous solution?

Problem:

- Need to integrate functions from different meshes
- Only becomes worse with multistep time stepping methods

Solution:

- This is clearly the “right thing to do”
- But it is not realistically feasible:
  – need to keep multiple meshes around
  – quadrature needs to find points on other meshes

- In practice we therefore always interpolate old solutions:

$$\left(\varphi_i^n, u_h^n\right) + \theta k_n \left(\nabla \varphi_i^n, \nabla u_h^n\right) = \left(\varphi_i^n, I_h^n u_h^{n-1}\right) + \ldots \qquad \forall i = 1 \ldots \dim V_h^n$$

How to treat the previous solution?

Practical approach:
Whenever we decide that we want to refine the mesh:

- refine mesh
- distribute DoFs on new mesh
- resize matrices and vectors
- interpolate all solution vectors from old to new mesh (using the *SolutionTransfer* class)
- build linear systems as if there had only ever been one mesh

When to refine the mesh?

Considerations:

- Ideally, refine in every time step
- But this is expensive

Compromise:

- Only every few time steps
- So that cost is not significant overall
- But so that mesh always resolves solution well
- For example, every 5–10 time steps

How to refine the mesh?

Considerations:

- For stationary problems, we can refine into singularities
- For time dependent problems, time step is always coupled somehow to *minimal* mesh size
- Refining into singularities makes time step very small

Compromise:

- Limit minimal mesh size to a certain refinement level
- In practice: also limit *maximal* mesh size
- Within these limits, mesh can change freely

How to choose the initial mesh?

Considerations:

- We haven't computed a solution yet

Approach 1:

- Interpolate/project initial solution
- Refine mesh based on $u^0(x)$
- Repeat

Approach 2:

- Interpolate/project initial solution
- Do one time step
- Refine mesh based on $u^1(x)$
- Repeat

Note: Choice of approach depends on problem.

MATH 676

–

Finite element methods in scientific computing

Wolfgang Bangerth, Texas A&M University