A Posteriori Based Adaptive Mesh Refinement

Computer Session B2

Instructions

✔ To pass this session you must present your solutions to all problems with this mark to the instructor.

✰ This mark indicates that the problem is also included in the course's problem demonstration sessions (gives bonus points).

Abstract Problem

In class we have studied the abstract variational problem: find $u \in V$ such that

$$a(u, v) = l(v), \quad \forall v \in V \tag{1}$$

where $a(\cdot, \cdot)$ is a bilinear form, $l(\cdot)$ is a linear form, and V is a Hilbert space with norm $\| \cdot \|_V$.

An example of a problem which can be recast into the abstract form (1) is the Poisson equation

$$-\Delta u = f, \quad \text{in } \Omega \tag{2a}$$

$$u = 0, \quad \text{on } \partial\Omega \tag{2b}$$

which yields the forms

$$a(u, v) = (\nabla u, \nabla v) \tag{3}$$

$$l(v) = (f, v) \tag{4}$$

on the Hilbert space $V = H_0^1(\Omega) = \{v : \|\nabla v\|^2 + \|v\|^2 < \infty,\ v|_{\partial\Omega} = 0\}$.

The existence of a solution to (1) is guaranteed by the Lax-Milgram lemma under the conditions that $a(\cdot,\cdot)$ is coercive, that is, there must exist a number $\alpha > 0$ such that

$$\alpha\|v\|_V^2 \leq a(v,v), \quad \forall v \in V \tag{5}$$

Further, $a(\cdot,\cdot)$ and $l(\cdot)$ must be continuous in the sense that

$$a(u,v) \leq C_1\|u\|_V\|v\|_V, \quad \forall v \in V \tag{6}$$
$$l(v) \leq C_2\|v\|_V, \quad \forall v \in V \tag{7}$$

for some positive constants C_i, $i = 1, 2$.

A finite element approximation to (1) is obtained by replacing V with a finite dimensional subspace $V_h \subset V$ typically consisting of piecewise polynomials on a mesh of the computational domain Ω. A prototype finite element method takes the form: find $u_h \in V_h$ such that

$$a(u_h, v) = l(v), \quad \forall v \in V_h \tag{8}$$

Problem 1. ✰ Show that the solution u to the abstract variational problem (1) satisfies the stability estimate $\|u\|_V \leq C/\alpha$. *Hint:* Use the coercivity and the continuity of the problem.

Problem 2. ✰ Assuming that the bilinear form $a(\cdot,\cdot)$ is symmetric, and satisfies $a(v,v) > 0$ with $a(v,v) = 0$ if and only if $v = 0$, it can be used to define a norm called the *energy norm* on V given by $\|\cdot\|_E = a(\cdot,\cdot)^{1/2}$. Show that the energy norm and $\|\cdot\|_V$ are equivalent norms, i.e. that there exist positive constants c_1 and c_2 such that $c_1\|v\|_V \leq \|v\|_E \leq c_2\|v\|_V$ for all $v \in V$.

Problem 3. ✰ Show that if u satisfies $a(u,v) = l(v)$ for all $v \in V$ then u also minimizes the functional $J(v) = \frac{1}{2}a(v,v) - l(v)$ on V. *Hint:* Write $v = u + w$ and show that $J(v) > J(u) + \dots$

Problem 4. ✰ The Poincaré inequality $\|v\| \leq C\|\nabla v\|$ holds for any sufficiently smooth function $v : \Omega \to \mathbb{R}$ provided that v is zero along some part of the boundary $\partial\Omega$. Use this to show that $\|\nabla v\|$ and $\|v\|_{H^1} = \|\nabla v\| + \|v\|$ are equivalent norms on $H_0^1(\Omega)$. In particular, verify that $\|\nabla v\| = 0$ implies $v = 0$ on $H_0^1(\Omega)$.

Problem 5. ✰ What numerical values do the constants α, C_1, and C_2 have for the problem $-\Delta u = xy^2$ on the square $\Omega = [-1, 2] \times [0, 3]$ assuming a zero boundary condition? *Hint:* The constant in the Poincaré inequality is $9/2$.

Problem 6. ✩ Consider the abstract variational problem (1) with

$$a(u, v) = v^T A u, \qquad l(v) = v^T b, \qquad V = R^n$$

where A is a real $n \times n$ matrix, b is a real $n \times 1$ vector, and $\| \cdot \|_V$ the usual Euclidean norm.

a) Show by a simple argument from linear algebra that there exist a unique solution $u \in V$ to (1) assuming that $a(\cdot, \cdot)$ is coercive on V.

b) Show that the coercivness of $a(\cdot, \cdot)$ is not really necessary in this case when V has finite dimension, and that it suffice that $a(v, v) > 0$.

Error Estimation

In order to get useful numerical results it is crucial to assert the accuracy of the finite element solution u_h by estimating the error $e = u - u_h$. A very important property of the error is that it satisfies the so-called Galerkin orthogonality

$$a(u - u_h, v) = 0, \quad \forall v \in V_h \tag{9}$$

which is readily obtained by subtracting (8) from (1).

In the mathematical literature there are two types of error estimates, namely, *a priori* and *a posteriori* estimates.

An a priori estimate uses the exact solution u to extract information about the error e. Since u is unknown this means that a priori estimates are not computable. However, a priori estimates can be used to deduce convergence rates of the finite element approximation, and thus determine if the finite element method will work or not.

The basic a priori estimate is known as Cea's lemma

$$\|u - u_h\|_V \leq \frac{C_1}{\alpha} \|u - v\|_V, \quad \forall v \in V_h \tag{10}$$

It says that u_h is closest to u of all functions in V when measured in the norm $\| \cdot \|_V$. The derivation of (10) goes as follows. Starting form the coercivity result we have

$$
\begin{aligned}
\alpha\|e\|_V^2 &\le a(e,e) && (11)\\
&= a(e, u-u_h) && (12)\\
&= a(e, u-v+v-u_h) && (13)\\
&= a(e, u-v)+0 && (14)\\
&\le C_1\|e\|_V\|u-v\|_V && (15)
\end{aligned}
$$

where we have first added and subtracted a $v \in V_h$, and then used the Galerkin orthogonality to deduce that $a(e, v-u_h) = 0$, since $v - u_h \in V_h$.

In contrast to a priori estimates, which use the exact solution u, a posteriori estimates use the finite element solution u_h to extract information about the error. As a consequence a posteriori estimates are computable and can be used to determine the quality of u_h for a specific problem. In particular a posteriori estimates can be used to detect regions where the error is large, thus indicating where u_h must be improved to give a better solution approximation. The most economic way of improving u_h is to make a local mesh refinement, which simply means inserting more nodes within the regions where the error e is large. This is called a posteriori based mesh refinement and gives rise to adaptive algorithms for the automatic enhancement of u_h.

A posteriori estimates are based on the following *error representation formula*

$$
\begin{aligned}
\alpha\|e\|_V &\le a(e,e) && (16)\\
&= a(e, e-v) && (17)\\
&= a(u, e-v) - a(u_h, e-v) && (18)\\
&= l(e-v) - a(u_h, e-v) && (19)\\
&= (R(u_h), e-v) && (20)
\end{aligned}
$$

where we have introduced the (weak) residual $(R(u_h), v) = l(v) - a(u_h, v)$, $\forall v \in V_h$.

Thus for the Poisson equation (2) we have the error representation formula

$$
\|\nabla e\|^2 \le (f, e-v) - (\nabla u_h, \nabla(e-v)) \qquad (21)
$$

To obtain a computable error estimate we must get rid of the error e occurring on the right hand side. To do so we break the integrals into a sum over the elements K of the mesh and integrate by parts

$$
\begin{aligned}
\|\nabla e\|^2 &\le \sum_K (f, e-v) - (\nabla u_h, \nabla(e-v)) && (22)\\
&= \sum_K (f, e-v) - (n\cdot\nabla u_h, e-v)_{\partial K} + (\Delta u_h, e-v) && (23)
\end{aligned}
$$

The reason for breaking up the integrals into a sum over the elements is that the normal derivative $n \cdot \nabla u_h$ is discontinuous over the element boundaries. Recall that for continuous piecewise linear finite elements ∇u is a constant vector on each element and thus there is a jump in the normal derivative when moving from one element to a neighboring. More specific, for two neighboring elements K^+ and K^- sharing edge E and with outward unit normal pointing from K^+ to K^- we define $[n \cdot \nabla U] = (\nabla U|_{K^-} - \nabla U|_{K^+}) \cdot n$. We then have

$$\sum_K -(n \cdot \nabla u_h, e - v)_{\partial K} = \sum_E ([n \cdot \nabla u_h], e - v)_E \tag{24}$$

In practice it is however convenient to express these integrals as a sum over the elements K and not over the edges E so we therefore redistribute the jump $[n \cdot \nabla u_h]$ equally between the two elements sharing E by writing

$$\sum_E ([n \cdot \nabla u_h], e - v)_E = \sum_K \tfrac{1}{2}([n \cdot \nabla u_h], e - v)_{\partial K} \tag{25}$$

To summarize we thus have

$$\|\nabla e\|^2 \leq \sum_{K \in \mathcal{K}} (f + \Delta u_h, e - v)_K + \sum_{K \in \mathcal{K}} \tfrac{1}{2}([n \cdot \nabla u_h], e - v)_{\partial K} \tag{26}$$

It remains to estimate these two sums. Choosing v as the interpolant $\pi e \in V_h$ to e we can estimate the first sum by using the interpolation estimate $\|v - \pi v\|_K \leq C h_K \|\nabla v\|_K$ and the Cauchy-Schwartz inequality

$$\sum_K (f + \Delta u_h, e - \pi e)_K \leq \sum_K h_K \|f + \Delta u_h\|_K h_K^{-1} \|e - \pi e\|_K \tag{27}$$

$$\leq \sum_K h_K \|f + \Delta u_h\|_K C \|\nabla e\|_K \tag{28}$$

where $h_K = \mathrm{diam}(K)$ is the size of element K. To estimate the second sum we again use the Cauchy-Schwartz inequality and the following variant of the so-called trace inequality $\|v\|_{\partial K} \leq C(h_K^{-1/2}\|v\|_K + h_K^{1/2}\|\nabla v\|_K)$.

$$\sum_K \tfrac{1}{2}([n \cdot \nabla u_h], e - \pi e)_{\partial K} \tag{29}$$

$$\leq \sum_K \tfrac{1}{2}\|[n \cdot \nabla u_h]\|_{\partial K} \|e - \pi e\|_{\partial K} \tag{30}$$

$$\leq \sum_K \tfrac{1}{2} h_K^{1/2} \|[n \cdot \nabla u_h]\|_{\partial K} C(h_K^{-1}\|e - \pi e\|_K + \|\nabla(e - \pi e)\|_K) \tag{31}$$

$$\leq \sum_K \tfrac{1}{2} h_K^{1/2} \|[n \cdot \nabla u_h]\|_{\partial K} C \|\nabla e\|_K \tag{32}$$

Combing these estimates we have

$$\|\nabla e\|^2 \leq C \sum_K (h_K \|f + \Delta u_h\|_K + \tfrac{1}{2} h_K^{1/2} \|[n \cdot \nabla u_h]\|_{\partial K}) \|\nabla e\|_K \tag{33}$$

$$\leq C \left(\sum_K h_K^2 \|f + \Delta u_h\|_K^2 + \tfrac{1}{4} h_K \|[n \cdot \nabla u_h]\|_{\partial K}^2 \right)^{1/2} \left(\sum_K \|\nabla e\|_K^2 \right)^{1/2} \tag{34}$$

$$\leq C \left(\sum_K h_K^2 \|f + \Delta u_h\|_K^2 + \tfrac{1}{4} h_K \|[n \cdot \nabla u_h]\|_{\partial K}^2 \right)^{1/2} \|\nabla e\| \tag{35}$$

Finally, dividing by $\|\nabla e\|$ we conclude that

$$\|\nabla e\| \leq C \left(\sum_K h_K^2 \|f + \Delta u_h\|_K^2 + \tfrac{1}{4} h_K \|[n \cdot \nabla u_h]\|_{\partial K}^2 \right)^{1/2} \tag{36}$$

which is our a posteriori estimate for Poisson's equation. The quantity η_K defined by

$$\eta_K^2 = h_K^2 \|f + \Delta u_h\|_K^2 + \tfrac{1}{4} h_K \|[n \cdot \nabla u_h]\|_{\partial K}^2 \tag{37}$$

is called the *element indicator* and can be used to identify the elements that contribute the most to the error. Note that η_K is computable since all involved quantities are known.

A Basic Adaptive Algorithm

The above line of reasoning leads us to formulate adaptive algorithms, which automatically can compute the finite element solution u_h to a desired level of accuracy, while at the same time using a minimum of computer resources. In doing so the main idea is to generate a sequence of solutions on successively finer mesh, at each stage selecting and refining those elements that are judged to contribute most to the error. The process is terminated either when a maximum number of elements is exceeded, or when each triangle contributes less than a preset tolerance, ϵ.

It is straightforward to translate Algorithm 1 into a Matlab code.

For simplicity we reuse the assembly routine from the previous computer session.

```
function MyAdaptivePoissonSolver(geom)
```

Algorithm 1 A Posteriori Based Adaptive Mesh Refinement

Choose a (coarse) initial mesh.
while the number of elements are not too many **do**
    Compute u_h.
    for all elements K **do**
        Compute the element indicator η_K defined by (37).
        if $\eta_K > \epsilon$ **then**
            Refine element K.
        end if
    end for
end while

```
% 1. Create initial mesh.
[p,e,t]=initmesh(geom,'hmax',1);
MAXNEL = 10000;
while size(t,2) < MAXNEL
  % 2. Compute finite element solution U.
  [A,R,b,r] = assemble(p,e,t);
  U = (A+R)\(b+r);
  figure(1)
  pdesurf(p,t,U)
  % 3. Evaluate element indicator.
  eta = pdejmps(p,t,...);
  % 4. Refine mesh.
  epsilon = 0.9*max(eta);
  [p,e,t] = refinemesh(geom,p,e,t,find(eta>=epsilon)');
  figure(2)
  pdemesh(p,e,t)
end
```

Here, the vector `eta` contains the element indicators so that entry number K of `eta` contains η_K. The actual evaluation of η_K is done by calling the build-in routine `pdejmps`

```
eta = pdejmps(p,t,c,a,f,U,1,1,1);
```

The reason for the complicated syntax is that the PDE-Toolbox is geared towards solving $-\nabla \cdot (c\nabla u) + au = f$. Hence, the three inputs `c`, `a`, and `f`. Each one of these can be given either as a constant, or as a row vector containing values of the coefficients c, a, and f at the element midpoints. In our case $c = 1$ and $a = 0$ so the variables `c=1` and `a=0` can be constants. However, f is a spatially varying function and thus `f` must be a vector of elemental mean values of f. For instance, if $f = x\sin(y)$ then we can compute `f` by writing

```
f = zeros(1,size(t,2));
i=t(1,:); j=t(2,:); k=t(3,:);
% compute element midpoints
x=(p(1,i)+p(1,j)+p(1,k))/3;
y=(p(2,i)+p(2,j)+p(2,k))/3;
% evaluate f(x,y)=x sin(y)
f=x.*sin(y);
```

After computing `f` we call `pdejmps` viz.

```
eta = pdejmps(p,t,1,0,f,U,1,1,1);
```

Further, the elements which have the largest indicators must be selected and refined. A commonly used refinement criteria is to refine element K if $\eta_K \geq \kappa \max_K \eta_K$, where the parameter $0 \leq \kappa \leq 1$ is a number which must be chosen by the user to indicate the portion of the elements to be refined. In Matlab, these elements are found by typing e.g.,

```
kappa = 0.9;
epsilon = kappa*max(eta);
find(eta>epsilon)'
```

Problem 7. ✔ Complete the code `MyAdaptivePoissonSolver` outlined above and solve adaptively

$$-\Delta u = \tfrac{1}{4}\,r^{-3/2}, \quad \text{in } \Omega \tag{38a}$$
$$u = 0, \qquad \text{on } \partial\Omega \tag{38b}$$

where $\Omega = \{r : 0 \leq r \leq 1\}$ is the unit circle with radius $r = \sqrt{x^2 + y^2}$. Verify that the analytic solution is $u = 1 - \sqrt{r}$. Calculate ∇u and $\|\nabla u\|$. Can you explain the tendency for the mesh refinement to cluster near origo?

The unit circle is a predefined geometry in Matlab and you can create a mesh of it by passing `'circleg'` to `initmesh`.

Tip: To avoid division by zero add 10^{-6} to r when computing f.

Problem 8. ✔ In the adaptive loop, draw the error indicator `eta` with `pdesurf`.

Problem 9. ✰ Verify the trace inequality $\|v\|_{\partial\Omega} \leq C(\|\nabla v\|^2 + \|v\|^2)^{\frac{1}{2}}$ for the particular choice $v = x$ on the square $\Omega = [0, L]^2$ with side length L. How does the constant C depend on L? How does this change if we instead consider $\|v\|_{\partial\Omega} \leq C(L\|\nabla v\|^2 + L^{-1}\|v\|^2)^{\frac{1}{2}}$?

Problem 10. ✔ Solve $-\Delta u = f$ on the square $\Omega = [-1, 1]^2$. Manufacture f so that the exact solution is $u = \exp(-c(x^2 + y^2))$ with $c = 100$, that is, so that u has the form of a narrow pulse centered at $(0, 0)$. You may set $u = 0$ on the boundary since u tend to zero quickly away from origo. Make a rough comparison of the accuracy of the computed solution when using adaptive mesh refinement as opposed to using uniform mesh refinement for the same amount of elements. Recall that making a uniform mesh refinement is the same as subdividing all elements, and not only those with large error contributions.

Problem 11. ✔

Consider a 'Pacman-shaped' domain Ω given by the code below. Solve $-\Delta u = 1$ on Ω with $u = 0$ on $\partial\Omega$. Plot the mesh and the numerical solution. What seems to be the source for refinement?

Hint: You might have to change the refinement criterion to guarantee some refinement at each iteration.

```
L=0.1;
geom=[1 sqrt(1-L^2) 0 L 1 1 0  0 0 1;
      1 0 -1 1 0 1 0  0 0 1;
      1 -1 0 0 -1 1 0  0 0 1;
      1 0 sqrt(1-L^2) -1 -L 1 0  0 0 1;
      2 sqrt(1-L^2) 0 -L 0 1 0  0 0 0;
      2 0 sqrt(1-L^2) 0 L 1 0  0 0 0]';
```