AMSC/CMSC 661 Scientific Computing II
Spring 2005
Solution of Elliptic PDEs
Part 2
Dianne P. O'Leary

These notes are based on the 2003 textbook of Stig Larsson and Vidar Thomée.

Solution and error estimates using finite differences (p. 46)

- Since the development of easy-to-use finite element packages, finite differences are becoming less popular:
  - Interpolation to find the solution at points other than mesh points is cumbersome.
  - The formulas need to be fixed up near the boundary.
  - For high order methods, 'near the boundary' is quite a large region.
- Finite differences still have some use for special problems such as Poisson's equation on a rectangle or box domain.

Recall:

$$
\begin{aligned}
u'(x) &= \frac{\mathbf{u(x+h) - u(x-h)}}{\mathbf{2h}} + O(h^2)\,, \\
u''(x) &= \frac{\mathbf{u(x-h) - 2u(x) + u(x+h)}}{\mathbf{h^2}} + O(h^2)
\end{aligned}
$$

for small values of h when $u \in \mathcal{C}^4$.

Unquiz 1: Consider the **homogeneous-Dirichlet** problem

$$\mathcal{A}u = -u_{xx} - u_{yy} + cu = f$$

on the domain $\Omega = (\mathbf{0}, \mathbf{1}) \times (\mathbf{0}, \mathbf{1})$, with $u = 0$ on the boundary. Assume that $c \geq 0$ is a constant. Let $h = 1/5$ ($M = 6$), and write the 16 finite difference equations for u at $x, y = .2, .4, .6$, and $.8$.

Properties of the finite difference formulation:

- We obtain a system of linear equations $AU = g$, where g is determined by the function f and the boundary conditions.
- A is $(M-1)^2 \times (M-1)^2$, symmetric, and **block tridiagonal with tridiagonal blocks**. In the jth row, the main diagonal element is $4/h^2 + c$ and the nonzero off-diagonal elements are $-1/h^2$.

- The matrix A is **row diagonally dominant**: the main diagonal element is at least as big as the sum of the absolute values of the off-diagonal elements. This ensures that the matrix **has no zero eigenvalues** and therefore a unique solution U exists.
- The finite difference approximation is formed from a **5-point stencil**.

Your book develops an error estimate (when $c = 0$) from a discrete maximum principle and a stability estimate, just as in the ODE-BVP case. Read pp. 46(bottom) - 49 if you are interested.

Solution and error estimates using finite elements (FE) (p. 57)

The plan:

- Derive the linear system of equations.
- Error analysis
- Practicalities

Deriving the FE system of equations

Notation: We consider this special case:

$$\mathcal{A}u = -\bigtriangledown \cdot (a \bigtriangledown u) = f \text{ in } \Omega \subset \mathcal{R}^2$$

with $u = 0$ on Γ.

Assumptions:

- $a(x)$ is a smooth function.
- $a(x) \geq \alpha > 0$ **in** $\bar{\Omega}$.
- $f \in L_2(\Omega)$.
- Ω is convex and Γ is a polygon.

Recall the **variational formulation**

$$a(u, v) = (f, v), \;\; v \in H_0^1$$

where

$$\begin{aligned} a(u, v) &= \int_\Omega (a \bigtriangledown u \cdot \bigtriangledown v) dx \\ (f, v) &= \int_\Omega f v dx \end{aligned}$$

For the ODE, we partitioned our domain into subintervals. Here, we partition Ω into a collection of **triangles** $\mathcal{T}_h$ so that

- $\bar{\Omega} = \bigcup_{K \in \mathcal{T}_h} K$.
- h_K is the diameter of the triangle K.
- $h = \max_{K \in \mathcal{T}_h} h_K$.
- P is the set of vertices of triangles in $\mathcal{T}_h$, and these are called the **nodes** of the **triangulation**. We number the **interior nodes**, those not on Γ, as $x_1, \ldots, x_M$.
- The triangulation is **admissible**, meaning that the intersection of any two triangles is either empty, a node, or an entire edge of both triangles. Picture (p. 58).
- We assume that the minimal angle in any of our triangles K is bounded below, independent of h. (Notice what this means if we decide to **refine the mesh**.)

We seek an approximate solution of a particular form:

- continuous,
- satisfying the boundary condition,
- and **piecewise linear** in each of the triangles K.

We call the space of such functions S_h and note that it is a subset of H_0^1, the space where the solution lives.

A convenient basis

We can construct our solution using **any** basis for S_h, but one basis is particularly convenient: the set of **hat functions** ϕ_i, $i = 1, \ldots, M$, designed to satisfy $\phi_i(x_i) = 1$ and $\phi_i(x_j) = 0$ if $i \neq j$.

Picture.

Any function $v \in S_h$ can be written as

$$v(x) = \sum_{i=1}^{M} v_i \phi_i(x)$$

where $v_i = v(x_i)$.

The resulting equations

Our original problem: Find $u \in H_0^1$ satisfying

$$a(u, v) = (f, v)$$

for all $v \in H_0^1$.

Our new problem: Find $u_h \in S_h$ satisfying

$$a(u_h, v) = (f, v)$$

for all $v \in S_h$.

Because the ϕ_i form a basis, our new problem becomes:

Find

$$u_h = \sum_{i=1}^{M} u_i \phi_i(x)$$

satisfying

$$a(u_h, \phi_j) = (f, \phi_j)$$

for $j = 1, \ldots, M$.

Unquiz 2: Write the resulting system of equations $AU = g$ for Poisson's equation using the particular triangulation ... and compare with the answer to Unquiz 1. []

Some properties

- This method of constructing the discrete equations is called **Galerkin's method** and is characterized by seeking u_h in some subspace of the space H_0^1 that contains the solution, and making the residual $a(u_h, v) - (f, v)$ zero on that subspace.
- A is called the **stiffness matrix** and g is called the **load vector**.
- A is symmetric (because $a(\phi_i, \phi_j) = a(\phi_j, \phi_i)$) and $V^T AV = a(v, v) > 0$ when
$$v = \sum_{i=1}^{M} v_i \phi_i(x) \neq 0.$$
Therefore, the solution exists and is unique.
- For the triangulation of the Unquiz, A is block tridiagonal with tridiagonal blocks. In general, A is sparse (since $a(\phi_i, \phi_j)$ is usually zero) but not quite this regular.

Some variations on our theme

- Higher order approximation
- Non-polygonal domains

Higher order approximation (p. 59)

We derived our finite element equation using the space of **piecewise linear functions** (i.e., piecewise polynomials of degree 1) with a convenient basis, the basis of **hat functions**.

We could also use **higher order polynomials**: quadratics, cubics, etc. The corresponding **hat functions** still vanish at all nodes but one, but their **support**, the domain over which they are nonzero, is now bigger.

We need more parameters (unknowns) to express our solution. For piecewise quadratics, for example, instead of just solving for the values u_i at the vertices of the triangles, we need extra nodes.

A quadratic function of two variables

$$[]x^2 + []xy + []y^2 + []x + []y + []$$

is determined by 6 coefficients, so we choose 6 nodes in each triangle; see picture on p. 59.

And we have a basis function for each node. At the vertices of the triangles we use our old hat functions. At the new interior nodes, we specify a quadratic function that vanishes at all nodes but one.

Picture: p. 57.

Non-polygonal domains

If Γ is not a polygon, then the triangulation does not fit Ω exactly.

Instead, we approximate it by a polygonal domain Ω_h.

For piecewise linear functions, nothing is lost.

For higher-order polynomial elements, care needs to be taken in order not to lose accuracy, but we won't discuss this.

FE error analysis

We will derive 3 kinds of error bounds:

- L_2 bounds.
- pointwise bounds.
- *a posteriori* bounds.

Deriving L_2 error bounds

The L_2 error analysis of the finite element method proceeds in two steps:

- **Step 1:** Show that for every function $u \in H_0^1$, there is a function $\hat{u}_h \in S_h$ that is close to it.
- **Step 2:** Show that the system of equations yields a solution close to $\hat{u}_h$.

Step 1: Approximability

For any $u \in H_0^1$, let $\hat{u}_h \in S_h$ be defined by

$$I_h u \equiv \hat{u}_h = \sum_{i=1}^{\mathbf{M}} u(x_i)\phi_i(x).$$

(This is the piecewise polynomial **interpolating function**.)

A standard result in approximation theory tells us that for **piecewise linear functions** over the triangle K we have

$$\begin{aligned} \|I_h u - u\|_K &\leq C_K h_K^2 |u|_{2,K}, \\ \| \bigtriangledown (I_h u - u)\|_K &\leq C_K h_K |u|_{2,K}. \end{aligned}$$

(The proof follows from Taylor series expansions (Bramble-Hilbert Lemma).)
(Remember notation: L_2 norm of u''.)

The constants C_K grow if the triangle gets too skinny, and that is why we made an assumption about the smallest angle.

So

$$\begin{aligned} \|I_h u - u\| &= \left(\sum_{K \in \mathcal{T}_h} \|I_h u - u\|_K^2 \right)^{1/2} \\ &\leq \left(\sum_K C_K^2 h_j^4 |u|_{2,K}^2 \right)^{1/2} \leq C h^2 \|u\|_2, \end{aligned}$$

for all $u \in H^2$ and similarly

$$|I_h u - u|_1 \leq C h \|u\|_2.$$

More generally, for **polynomials of degree $r-1$** (p. 61), for all $u \in H^r$,

$$
\begin{aligned}
\|I_h u - u\|_K &\leq Ch_K^r |u|_{r,K}, \\
\|I_h u - u\| &\leq Ch^r \|u\|_r, \\
\| \bigtriangledown (I_h u - u)\|_K &\leq Ch_K^{r-1} |u|_{r,K}, \\
|(I_h u - u)|_1 &\leq Ch^{r-1} \|u\|_r.
\end{aligned}
$$

Note: These bounds are only of interest if the triangulation is approximately uniform, with all $h_K \approx h$.

Step 2: u_h is close to $I_h u$

We use the **energy norm**

$$\|v\|_a = a(v, v)^{1/2}.$$

Theorem 5.3a (p. 63):

$$(**)\ \|u_h - u\|_a = \min_{v \in S_h} \|v - u\|_a$$

A note: Let $e = u - u_h$. We know that $a(u, v) = (f, v)$ and $a(u_h, v) = (f, v)$ for all $v \in S_h$, so

$$(***)\ a(e, v) = 0$$

for all $v \in S_h$. This means that the **error is orthogonal to S_h**, or, in other words, u_h is the **orthogonal projection** (with respect to the inner product a) of u onto S_h, and therefore $(**)$ holds.

Theorem 5.3b (p. 63):

$$|\mathbf{u_h} - \mathbf{u}|_1 \leq Ch\|u\|_2.$$

This is nice, but it gives us a result on the **energy norm**, not the L_2 norm, so we need to work a little more.

Theorem 5.4 (p. 64):

$$\|e\| \leq Ch^2 \|u\|_2.$$

Compare these three results with pp. 54-55.

Deriving pointwise error bounds

The error bound we just stated is error in an **average** sense, integrated over Ω. Sometimes we need a **pointwise error bound**:

Theorem: (p. 66) Assume, in addition to our previous assumptions, that $h_K \geq ch$ for all triangles K, where c is a constant independent of h. **(In other words, no triangle is much smaller than any other.)** Then for h sufficiently small,

$$\|u_h - u\|_{\mathcal{C}} \leq Ch^2 \log(1/h) \|u\|_{\mathcal{C}^2}.$$

Deriving a posteriori error bounds

A major disadvantage of the error bounds we have so far: they all include the norm of the unknown solution u!

Therefore, they are **not computable**.

An **a posteriori error bound** is one that can be computed from the approximate solution u_h.

Theorem 5.6: (p. 66) Let

$$R_K = h_k^2 \|\mathcal{A}u_h - f\|_K + h_K^{3/2} \|a(n \cdot \bigtriangledown u_h)\|_{\Gamma(K) - \Gamma(\Omega)},$$

where $n \cdot \bigtriangledown u_h$ denotes the jump of the normal derivative of u_h across the boundary of K. Then

$$\|u - u_h\| \leq C \left(\sum_K R_K^2 \right)^{1/2}.$$

Proof: Using duality; see p. 66.

FE practicalities

In order to compute with the FE method, we need to do these things:

- Triangulate Ω.
- Assemble the stiffness matrix and the right-hand side.
- **Solve the linear system.**
- Estimate the error.
- Refine the mesh if necessary.
- Compute quantities of interest.

In 3-d, we face the same issues, but (of course) it gets more complicated. We'll stick to 2-d here.

FE triangulation

Recall the properties that we need:

- If two triangles intersect, their intersection must be a vertex or an entire edge.
- The triangles must not be too "skinny".
- If Γ is not a polygon, the triangles must still hug the boundary closely.

Input: A description of the domain Ω:

- a list of vertices of a polygonal domain, or
- software defining the boundary as a function of $\theta \in [0, 2\pi]$, or
- a coarse triangulation of Ω.

plus some indication of how fine a mesh is desired.

Output: A data structure containing the triangles, coordinates of the nodes, and adjacency information.

As you might imagine, **this software is not easy to write!**

Pointers to mesh generation software and the research community:
http://www-users.informatik.rwth-aachen.de/~roberts/meshgeneration.html

Matlab has a mesh generator that we will use.

FE assembly of the stiffness matrix

(We also need the right-hand side, but the issues are similar.)

Recall that the (i, j) element of the stiffness matrix is

$$\begin{aligned} a(\phi_i, \phi_j) &= \int_\Omega a \bigtriangledown \phi_i(x) \cdot \bigtriangledown \phi_j(x) dx \\ &= \sum_K \int_K a \bigtriangledown \phi_i(x) \cdot \bigtriangledown \phi_j(x) dx \end{aligned}$$

and that almost all of the integrals in the summation are zero.

IMPORTANT Practical Note: If we are not careful, then we will end up computing n_T integrals for each matrix entry (where n_T is the number of triangles), making the work of assembling the matrix $O(M^3)$ (where M is the number of nodes).

If we **are** careful, the work will be just $O(M)$.

Efficient matrix assembly

Often, we take the data structure returned by the mesh generator and proceed triangle by triangle.

In each triangle, we compute all nonzero terms

$$a_K(\phi_i, \phi_j) = \int_K a \bigtriangledown \phi_i(x) \cdot \bigtriangledown \phi_j(x) dx$$

by making use of the adjacency information.

We could store the terms in the data structure for the triangulation. This would be quite sufficient if we plan to use an **iterative method** like **conjugate gradients** for solution of the linear system of equations, since all we need to do is to form matrix-vector products.

If we plan to use a **direct method** like **Cholesky factorization**, then we need a more explicit representation of the matrix. Again, if we are not careful, we will do too much work: there are $O(M^2)$ entries in the matrix, but only $O(M)$ of them are nonzero.

So as we compute the nonzero terms, we assign a storage location to the (i, j) matrix entry and add the term into it.

As a result, we would have an array of row indices, column indices, and values of the $O(M)$ nonzero elements.

Actually computing the integrals in the terms

We need to compute terms of the form

$$a_K(\phi_i, \phi_j) = \int_K a \bigtriangledown \phi_i(x) \cdot \bigtriangledown \phi_j(x) dx.$$

Unless the problem is very simple, we will not be able to do this **exactly**.

Instead, we will need to use **techniques of numerical integration**. (See 660!)

We will approximate by

$$a_K(\phi_i,\phi_j) \approx \sum_{\ell=1}^{l} w_{\ell,K} a \bigtriangledown \phi_i(x_\ell) \cdot \bigtriangledown \phi_j(x_\ell).$$

where x_ℓ is a point in K and $w_{\ell,K}$ are the weights of the integration formula.

Very important warning:

- The use of numerical integration means that we are not solving the finite element problem; instead, we are solving some approximation to it.
- Stiffness matrices tend to be **ill-conditioned**, meaning that small changes in the data can make large changes in the computed solution.
- Therefore, the numerical integration formula needs to have error small enough that the change in the answer is no larger than the error we expect due to the finite element approximation.

Example: For piecewise linear finite elements, we can use a **barycentric integration rule**, approximating

$$I_K(s) \equiv \int_K s(x)dx \approx area(K)s(P) \equiv q_K(s)$$

where P is the average of the three vertices of the triangle, i.e., the barycenter of the triangle. The error formula is

$$|I_K(s) - q_K(s)| \le C h_K^2 |s|_{W_1^2(K)}$$

where the semi-norm $|s|_{W_1^2(K)}$ is defined on p. 68.

Using error bounds like this, we can show that we solve a FE approximation to a problem **close** to the given problem, and, therefore, since the elliptic equation is stable, the solutions are close to each other.

See pp. 68-71 for details.

Notice that it is the **theory** that bails us out when we have to make these errors in computation.

FE error estimation

Recall **Theorem 5.6:** (p. 66) Let

$$R_K = h_k^2 \|\mathcal{A}u_h - f\|_K + h_K^{3/2} \|a(n \cdot \bigtriangledown u_h)\|_{\Gamma(K)-\Gamma(\Omega)},$$

where $n \cdot \bigtriangledown u_h$ denotes the jump of the normal derivative of u_h across the boundary of K. Then

$$\|u - u_h\| \leq C \left(\sum_K R_K^2 \right)^{1/2} .$$

We performed our FE calculation for some mesh with parameter h, and after we compute our solution, we can form the error estimate

$$\|u - u_h\| \leq C \left(\sum_K R_K^2 \right)^{1/2} .$$

What if it is bigger than we need it to be?

In that case we need to refine the mesh.

FE mesh refinement

Two possible refinements:

- Divide each triangle into 4. **Why not just bisect every triangle?** This gives us a matrix 4 times as big, and every nonzero matrix entry needs to be recomputed!
- Confine the refinement to the region where R_K is large.
  - This gives a much smaller increase in the size of the matrix.
  - But we need to be careful to keep the triangulation admissible and gradual.

FE computation of quantities of interest

Our computation is not finished when we solve the linear system and compute an acceptable error estimate.

What still might need to be done:

- Display the solution graphically.
- Compute the maximum value of the solution (easy for piecewise linear elements, more computationally expensive for higher order elements).
- Compute the energy norm of the solution.

- Compute derivatives of the solution. **But this has lower accuracy than our estimate of the solution.** If we really need this, we might want to use a **mixed FE method**, in which the derivative is approximated directly rather than derived from u. (See Section 5.7 if you are interested.)

- . . ..

Summary

- We have shown existence, uniqueness, and stability of the solution to our elliptic PDE.

- We have introduced several tools for analysis, including
  - approximability,
  - duality,
  - the energy norm,
  - regularity.

- We have defined a finite difference approximation to the elliptic PDE, reducing the problem to solving a linear system of equations.

- We have defined a finite element approximation.

- We showed existence and uniqueness of the finite element approximation, as well as an error bound.

- We have discussed the practicalities of finite element implementation, except for **efficient solution of the resulting system of linear equations**.