UW–Madison Math/CS 714

Methods of Computational Mathematics I

Elliptic equations II

Instructor: Yue Sun (yue.sun@wisc.edu)

September 18, 2025

Accuracy and stability

The truncation error τ_{ij} at (x_i, y_j) is defined by substituting the true solution $u(x, y)$ into the discretization and looking at what is left:

$$\tau_{ij} = \frac{1}{h^2}\big(u(x_{i-1}, y_j) + u(x_{i+1}, y_j) + \\ + u(x_i, y_{j-1}) + u(x_i, y_{j+1}) - 4u(x_i, y_j)\big) - f(x_i, y_j).$$

Using Taylor series, this is

$$\tau_{ij} = u_{xx} + u_{yy} - f(x_i, y_j) + \frac{h^2}{12}(u_{xxxx} + u_{yyyy}) + O(h^4),$$

and substituting in the original equation shows that

$$\tau_{ij} = \frac{h^2}{12}(u_{xxxx} + u_{yyyy}) + O(h^4).$$

Global error

The global error is defined as $E_{ij} = u_{ij} - u(x_i, y_j)$. It solves the linear system

$$A^h E^h = -\tau^h$$

where $\tau^h \in \mathbb{R}^{m^2}$ are the local errors assembled into a vector, and $A^h \in \mathbb{R}^{m^2 \times m^2}$ is the linear system describing the discretization. As before, the superscript h indicates that the quantities are associated with a mesh size h.

We aim to show that $\|(A^h)^{-1}\|$ is uniformly bounded at $h \to 0$. This requires finding the eigenvalues and eigenvectors of A^h.

Finding the eigenvalues of A^h

To find the eigenvalues of A^h, we can connect our current two-dimensional problem back to the eigenvalues of the one-dimensional BVP that we considered previously.

For the one-dimensional BVP, the eigenvectors were

$$u_i^p = \sin p\pi ih$$

for $p = 1, \ldots, m$, with corresponding eigenvalues

$$\lambda_p = \frac{2(\cos p\pi h - 1)}{h^2}.$$

Finding the eigenvalues of A^h

For the Poisson problem, the matrix can be split into $A^h = A^{h,x} + A^{h,y}$ where

$$(A^{h,x}u)_{ij} = \frac{u_{i-1,j} - 2u_{i,j} + u_{i+1,j}}{h^2},$$
$$(A^{h,y}u)_{ij} = \frac{u_{i,j-1} - 2u_{i,j} + u_{i,j+1}}{h^2}.$$

Consider the candidate eigenvector

$$u_{i,j}^{p,q} = (\sin p\pi ih)(\sin q\pi jh).$$

By comparing to the one-dimensional BVP

$$(A^{h,x}u^{p,q})_{i,j} = \lambda_p(\sin p\pi ih)(\sin q\pi jh) = \lambda_p u_{i,j}^{p,q}$$

and

$$(A^{h,y}u^{p,q})_{i,j} = (\sin p\pi ih)\lambda_q(\sin q\pi jh) = \lambda_q u_{i,j}^{p,q}.$$

Finding the eigenvalues of A^h

It follows that

$$(A^h u^{p,q})_{i,j} = (\lambda_p + \lambda_q) u^{p,q}_{i,j}$$

and therefore $u^{p,q}$ is an eigenvector with eigenvalue

$$\lambda_{p,q} = \lambda_p + \lambda_q = \frac{2(\cos p\pi h + \cos q\pi h - 2)}{h^2}.$$

All eigenvalues are negative, and the one closest to zero is

$$\lambda_{1,1} = -2\pi^2 + O(h^2).$$

The spectral radius (which is also the 2-norm) is therefore

$$\rho((A^h)^{-1}) = \left| \frac{1}{\lambda_{1,1}} \right| \approx \frac{1}{2\pi^2},$$

so the discretization is stable.

Condition number

We can also compute the condition number[1] of the matrix. This will be useful later.

The 2-norm of A^h is determined by eigenvalue of largest magnitude, so that $\|A^h\| = |\lambda_{m,m}| \approx 8/h^2$. The condition number is

$$\kappa(A^h) = \|A^h\| \|(A^h)^{-1}\| \approx \frac{8}{2\pi^2 h^2} = \frac{4}{\pi^2 h^2} = O(h^{-2}).$$

Thus the matrix becomes ill-conditioned as the grid is refined.

[1] See homework 0 question 4, Harvard AM205 video 0.3, and associated notes

Code example #2

The program *poisson2.py* solves the Poisson equation using the method of manufactured solutions with

$$u_{\text{manu}}(x, y) = x(x-1)e^{xy} \sin \pi y$$

corresponding to

$$f(x, y) = -e^{xy} \big(2\pi(x-1)x^2 \cos \pi y + (2 - 2y + x((x-1)(x^2 + y^2 - \pi^2) + 4y)\big) \sin \pi y$$

and boundary conditions $u = 0$ on $\partial\Omega$. It measures the global error for a variety of grid sizes.

Code example #2

The global error $E = u - \hat{u}$ in both the 2-norm and infinity norm is $O(h^2)$.

m	h	$\|E\|_2$	$\|E\|_\infty$
7	$\frac{1}{8}$	7.81×10^{-3}	1.65×10^{-3}
15	$\frac{1}{16}$	1.94×10^{-4}	4.10×10^{-4}
31	$\frac{1}{32}$	4.85×10^{-5}	1.02×10^{-4}
63	$\frac{1}{64}$	1.21×10^{-5}	2.56×10^{-5}
95	$\frac{1}{96}$	5.39×10^{-6}	1.14×10^{-5}

9-point Laplacian

Another discretization for the Laplacian is based on nine points, so that

$$(\nabla_9^2 u)_{ij} = \frac{1}{6h^2}\big(4(u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1}) \\ + u_{i-1,j-1} + u_{i-1,j+1} + u_{i+1,j-1} + u_{i+1,j+1} - 20u_{ij}\big).$$

Taylor expanding shows that

$$\nabla_9^2 u(x_i, y_j) = \nabla^2 u + \frac{h^2(u_{xxxx} + 2u_{xxyy} + u_{yyyy})}{12} + O(h^4).$$

The leading order error terms are larger than for the five-point stencil. But these additional terms allow the error to be written as

$$\nabla_9^2 u(x_i, y_j) = \nabla^2 u + \frac{h^2\nabla^2(\nabla^2 u)}{12} + O(h^4).$$

The operator $\nabla^2\nabla^2$ is written as ∇^4 and is called the biharmonic operator.

A more accurate method

This special feature of the 9-point stencil can be used to improve the accuracy of the method. Since $\nabla^2 u = f$,

$$\nabla^2(\nabla^2 u) = u_{xxxx} + 2u_{xxyy} + u_{yyyy} = \nabla^2 f$$

If $f = 0$, then $\nabla^2(\nabla^2 u) = 0$ and therefore

$$\nabla_9^2 u(x_i, y_j) = \nabla^2 u + \frac{h^2 \nabla^2(\nabla^2 u)}{12} + O(h^4) = \nabla^2 u + O(h^4).$$

Therefore solutions to the Laplace equation will be fourth-order accurate.[2]

[2] Assuming that the 9-point discretization is stable.

A more accurate method

The program *laplace.py* implements the 9-point stencil for the Laplace equation using the manufactured solution

$$u_{\mathrm{manu}}(x, y) = \cos 3x \exp 3y$$

Dirichlet boundary conditions $u = u_{\mathrm{manu}}$ are applied on $\partial\Omega$.

The program computes the global error for a variety of grid sizes.

A more accurate method

The global error $E = u - \hat{u}$ in both the 2-norm and infinity norm is shown in the table below.

m	h	$\|E\|_2$	$\|E\|_\infty$
7	$\frac{1}{8}$	5.56×10^{-7}	1.30×10^{-6}
15	$\frac{1}{16}$	8.73×10^{-9}	2.03×10^{-8}
31	$\frac{1}{32}$	1.37×10^{-10}	3.22×10^{-10}
63	$\frac{1}{64}$	2.17×10^{-12}	5.12×10^{-12}
95	$\frac{1}{96}$	1.92×10^{-13}	4.64×10^{-13}

The errors scale like h^6, which is better than $O(h^4)$ that we expected. This is still consistent with our analysis, but likely means that the leading order error term is also canceling.

Extension to the Poisson equation

The fourth-order method can be extended to the Poisson equation $\nabla^2 u = f$ where f is non-zero. The numerical f_{ij} is modified to

$$f_{ij} = f(x_i, y_j) + \frac{h^2}{12}\nabla^2 f(x_i, y_j)$$

and the additional term cancels out the leading order error. The program *poisson3.py* demonstrates this using the example of

$$u(x, y) = x^3(1 - x)y(1 - y),$$

which has

$$f(x, y) = 2(x - 1)x^3 + 6x(2x - 1)(y - 1)y$$

and

$$\nabla^2 f(x, y) = 24(x(2x - 1) + (y - 1)y).$$

Extension to the Poisson equation

The program *poisson3.py* confirms that the method has $O(h^2)$ errors without the adjustment, and $O(h^4)$ errors with the adjustment.

Rather than compute $\nabla^2 f$ analytically, it is also possible to compute it numerically.

The code also demonstrates this, using a 5-point Laplacian stencil for $\nabla^2 f$, and again achieving $O(h^4)$ error.

There are other examples of methods like this, where an additional term is incorporated to cancel out leading-order error.

Other examples of elliptic equations

In the heat conduction example, the conductivity $\kappa(x, y)$ may be spatially varying. This results in the equation

$$\nabla \cdot (\kappa \nabla u) = f.$$

The same equation appears in other situations. In porous media flow,[3] through a medium with spatially varying permeability κ, the fluid pressure p satisfies

$$\nabla \cdot (\kappa \nabla p) = f$$

where in this case f represents fluid inflow and outflow.

Nonlinear elliptic equations also arise, and can be solved using similar methods (*e.g.* Newton) as the one-dimensional BVP.

[3] See, *e.g.*, N. J. Derr *et al.*, *Flow-driven branching through a porous medium*, Phys. Rev. Lett. **125**, 158002 (2020). (doi:10.1103/PhysRevLett.125.158002)