Lab Assignment 10
Finite Difference Method for the 2D Poisson Equation

ACS II
Spring 2020

Assigned March 12th, 2020
Due March 26th, 2020

Introduction

In this lab exercise, we consider the 2D Poisson equation,

$$\Delta u(x, y) \;=\; -f(x, y), \tag{1}$$

on the domain $\Omega = [0, 1] \times [0, 1]$ with boundary $\partial\Omega$. To start with we will consider problems with Dirichlet boundary conditions, i.e. $u(x, y) = g(x, y)$ on $\partial\Omega$.
Let us partition the domain into an $n \times m$ grid with spacing $\Delta x = 1/n$ and $\Delta y = 1/m$, and let $f_{i,j} = f(x_i, y_j)$ and $U_{i,j} \approx u(x_i, y_j)$. We can discretize the second derivatives using a second-order central difference scheme:

$$\frac{\partial^2 U_{i,j}}{\partial x^2} \approx \frac{U_{i-1,j} - 2U_{i,j} + U_{i+1,j}}{(\Delta x)^2},$$
$$\frac{\partial^2 U_{i,j}}{\partial y^2} \approx \frac{U_{i,j-1} - 2U_{i,j} + U_{i,j+1}}{(\Delta y)^2}.$$

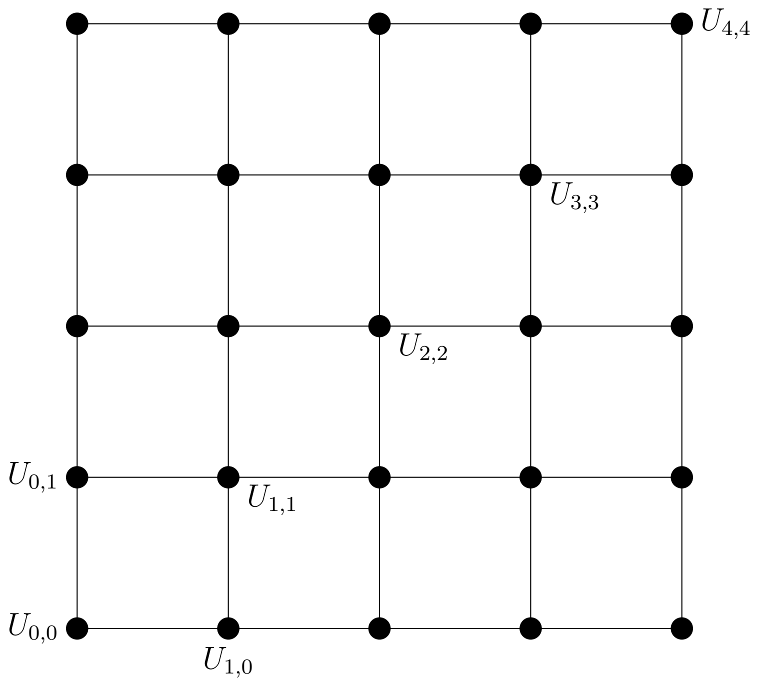
If we use these approximations in the 2D Poisson equation, we obtain

$$\frac{U_{i-1,j} - 2U_{i,j} + U_{i+1,j}}{(\Delta x)^2} + \frac{U_{i,j-1} - 2U_{i,j} + U_{i,j+1}}{(\Delta y)^2} = -f_{i,j}.$$

Assuming $\Delta x = \Delta y = h$ this simplifies to:

$$U_{i-1,j} + U_{i+1,j} + U_{i,j-1} + U_{i,j+1} - 4U_{i,j} = -h^2 f_{i,j}. \tag{2}$$

As an example consider solving equation (1) with 5 nodes in each direction.



At the node (x_2, y_2) we can apply (2) to get:

$$U_{1,2} + U_{3,2} + U_{2,1} + U_{2,3} - 4U_{2,2} = -h^2 f_{2,2}.$$

At the node (x_1, x_2) applying (2) gives:

$$U_{0,2} + U_{2,2} + U_{1,1} + U_{1,3} - 4U_{1,2} = -h^2 f_{1,2}.$$

Note that $U_{0,2}$ is on the boundary. This means its value is known from the boundary condition g and can be moved to the right hand side:

$$U_{2,2} + U_{1,1} + U_{1,3} - 4U_{1,2} = -h^2 f_{1,2} - g_{0,2}.$$

Proceeding in this way, we could construct 9 equations for the 9 interior nodes where our solution is unknown. In general (for larger problems) this will be a sparse, banded system. This system could be solved efficiently using standard linear algebra packages.
In this lab however, we will use the Jacobi method to solve the system. The Jacobi method is nice for this problem because it is simple to implement and does not require us to construct a matrix. In addition it allows us to keep using the simpler index notation above instead of renumbering our nodes from to 0 to $(n-1)^2$. The Jacobi method proceeds as follows:

1. Start with an initial guess to the solution, perhaps $U^0_{i,j} = 0$ for all i and all j
2. From U^0, compute $U^1_{i,j}$ for all i and j according to (2). This is:

$$U^1_{i,j} = \frac{U^0_{i-1,j} + U^0_{i+1,j} + U^0_{i,j-1} + U^0_{i,j+1} + h^2 f_{i,j}}{4}.$$

   For nodes bordering a boundary this equation will be modified slightly.
3. Check $||U^1 - U^0||$ (in the Frobenius norm). If this is smaller than some tolerance stop, otherwise set U^0 to U^1 and go back to step 1.

Experiments

This lab must be completed in a compiled language. All plots may be done in Matlab or Python.

1. Consider the problem (1) with $f(x,y) = -(20x^3y^3 + 6x^5y)$ and the boundary condition $g(x,y)$ on the unit box defined as follows:

$$g(x,y) = \begin{cases} 0 & \text{if } x = 0 \text{ or } y = 0 \\ x^5 & \text{if } y = 1 \\ y^3 & \text{if } x = 1 \end{cases}$$

Compute a finite difference a finite difference approximation to $u(x,y)$ with $h = 1/4$, 1/8 and 1/16. Use a tolerance of 1×10^{-5} for the Jacobi solver. The exact solution to this problem is $u(x,y) = x^5y^3$. For each h do the following:

(a) compute the error in the approximation according to:

$$e = h||U_{i,j} - u(x_i, y_j)||,$$

where $|| \cdot ||$ is the Frobenius norm.

(b) plot the residuals $||U^1 - U^0||$ as a function of the Jacobi iteration

What do you notice about the errors? In particular how fast does it decrease as we halve h?

2. Consider a problem with periodic boundary conditions in the x direction:

$$\begin{aligned} u(x,y) &= \begin{cases} 0 & \text{if } y = 0 \\ \sin^2(\pi x) & \text{if } y = 1 \end{cases} \\ u(0,y) &= u(1,y) \\ u_x(0,y) &= u_x(1,y) \\ f(x,y) &= 0 \end{aligned}$$

Solving this problem requires only minor modifications to your code. In particular instead of solving for $U_{i,j}$ at only the interior nodes, we now have to solve for $U_{i,j}$ on *either* the right boundary or the left boundary (periodicity means the solution on the left and right boundary must be equal). Let's compute the solution on the left boundary. This means we will need to find $U_{0,j}$ for all j. If we apply the stencil (2) at a node on the left boundary we get:

$$\frac{U_{-1,j} - 2U_{0,j} + U_{1,j}}{(\Delta x)^2} + \frac{U_{0,j-1} - 2U_{0,j} + U_{0,j+1}}{(\Delta y)^2} = -f_{0,j}.$$

This is problematic because we need $U_{-1,j}$ which doesn't exist! However we can invoke periodicity to say that $U_{-1,j} = U_{n-2,i}$. Likewise near right boundary we will need $U_{n+1,j}$, which is equal to $U_{0,j}$.

Compute a finite difference approximation to this problem for $h = 1/4$, $1/8$ and $1/16$. Use a tolerance of 1×10^{-5} for the Jacobi solver. For each h do a contour plot of your solution as well as a plot of the Jacobi residuals vs iteration.

Submission and Grading

To get credit for this assignment, you must upload the following information to canvas by 11:59pm, March 26th, 2020

- your source code files
- Your report as a single file in pdf format, including results from your work and relevant discussion of your observations, results, and conclusions.

This information must be received by 11:59pm, March 26th, 2020. As stated in the course syllabus, late assignment submissions will be subject to a 10% point penalty per 24 hours past the due date at time of submission, to a maximum reduction of 50%, according to the formula:

*[final score] = [raw score] - min(0.5, 0.1 * [# of days past due]) * [maximum score]*