Methods of matrix inversion

November 28, 2016

In this problem sheet, we will consider Laplace's equation on a square.

$$u_{xx} + u_{yy} = 0 \qquad (x, y) \in [0, 1] \times [0, 1] \tag{1}$$

We will deliberately take trivial boundary conditions so that the solution can be found analytically. This allows us to focus on the coding of different methods, and provides a good check for our results. The boundary conditions are

$$\begin{aligned} u(x, 0) &= 0 \\ u(0, y) = u(1, y) &= y \\ u(x, 1) &= 1 \end{aligned} \tag{2}$$

which leads to the solution $u(x, y) = y$.

We will start by considering a 4×4 grid. This means we will have 4 unknowns (boundary values are known) and must take $h = 1/3$. The resulting system of equations can be written (see class for derivation) as

$$\begin{bmatrix} -4 & 1 & 1 & 0 \\ 1 & -4 & 0 & 1 \\ 1 & 0 & -4 & 1 \\ 0 & 1 & 1 & -4 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} -h \\ -h \\ -1 - 2h \\ -1 - 2h \end{bmatrix}. \tag{3}$$

One can manually checked that this is solved by $u_1 = u_2 = h$, $u_3 = u_4 = 2h$.

Now that we have written our equations in matrix form, we can forget about the context and consider the problem at a more abstract level. The question now is how to quickly and accurately solve the system of equations

$$A\boldsymbol{u} = b \tag{4}$$

Exercise 1.
For each of the following methods derive an abstract, matrix form of the iteration and then write code that implements it. At each step, your code should calculate the average value of the solution and stop when the change in this value between consecutive steps is below some amount `tol`.

1. Point-Jacobi
2. Gauss-Seidel
3. SOR method with $w = 1.5$

Check that all of your methods produce the correct solution for u.

Add counters to your code to record out how many iterations each method requires. Play around with the over-relaxation parameter w – this drastically affects the speed of the SOR method. Can you find an optimal value for w? Does this agree with the literature?

Exercise 2.
For larger grids, these methods become much more useful. Derive a general form for the matrix A and write a function that takes an input N and creates a matrix that corresponds to an $N \times N$ grid. Test your code using $N = 4$.

The vector b depends on boundary conditions, so is generally part of the data for the problem. With these boundary conditions, what is the form for general N? Try your different iterative codes on a larger grid. What is the optimal value for w in this case?