Homework #3

Sam Fleischer

November 11, 2016

Problem 1

Use Jacobi, Gauss-Seidel, and SOR (with optimal ω) to solve

$$\nabla^2 u = -\exp\left(-(x-0.25)^2 - (y-0.6)^2\right)$$

on the unit square $(0,1) \times (0,1)$ with homogeneous Dirichlet boundary conditions. Find the solution for mesh spacings of $h = 2^{-5}$, 2^{-6}, and 2^{-7}. What tolerance did you use? What stopping criteria did you use? What value of ω did you use? Report the number of iterations it took to reach convergence for each method for each mesh.

I used the following iterative methods to solve

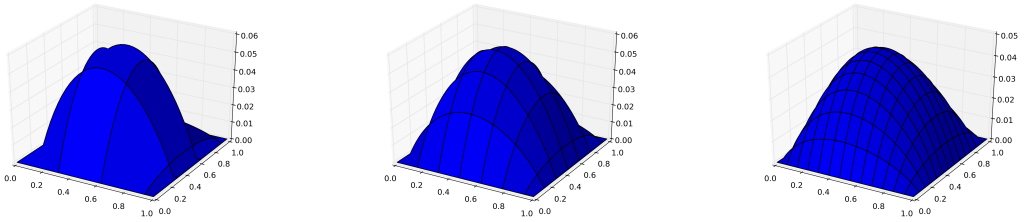
$$\nabla^2 u = -\exp\left(-(x-0.25)^2 - (y-0.6)^2\right)$$

on the unit square with grid size 2^{-i} for $i = 5, 6, 7$:

- Jacobi:

$$u_{i,j}^{k+1} = \frac{1}{4}\left(u_{i-1,j}^{k} + u_{i+1,j}^{k} + u_{i,j-1}^{k} + u_{i,j+1}^{k} - h^2 f_{i,j}\right)$$

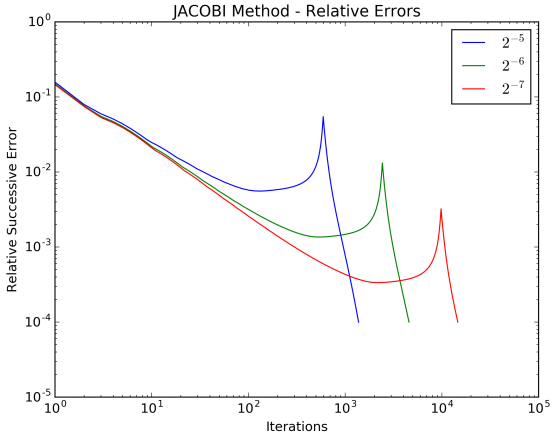
  Here are the results for $h = 2^{-i}$ for $i = 5, 6, 7$.



The number of iterations required to have a relative error less than 0.0001 (that is, $\left\|u^{k+1} - u^k\right\|_1 < \varepsilon \|u_k\|_1$, where $\varepsilon = 0.0001$) is

h	iterations	multiplicative factor	time taken (in seconds)
2^{-5}	1377		3.549528
2^{-6}	4570	3.32	44.084062
2^{-7}	14607	3.20	575.336392

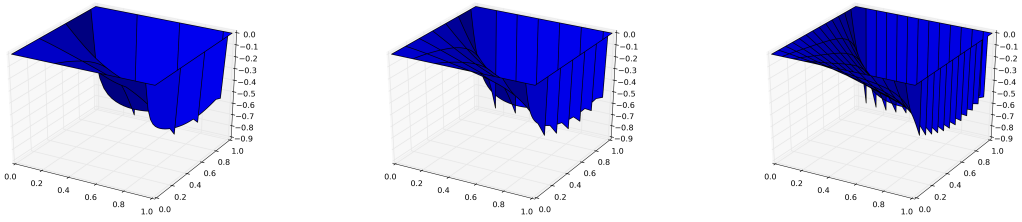
Here is a graph of the relative errors as a function of the iteration number:



- Gauss-Seidel

$$u_{i,j}^{k+1} = \frac{1}{4}\left(u_{i-1,j}^{k+1} + u_{i+1,j}^{k} + u_{i,j-1}^{k+1} + u_{i,j+1}^{k} - h^2 f_{i,j}\right)$$

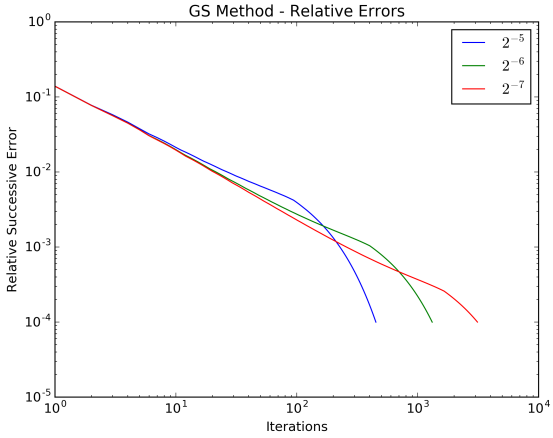
Here are the results for $h = 2^{-i}$ for $i = 5, 6, 7$.



The number of iterations required to have a relative error less than 0.0001 (that is, $\left\|u^{k+1} - u^{k}\right\|_1 < \varepsilon \|u_k\|_1$, where $\varepsilon = 0.0001$) is

h	iterations	multiplicative factor	time taken (in seconds)
2^{-5}	452		1.144104
2^{-6}	1319	2.92	13.163329
2^{-7}	3121	2.37	123.097511

Here is a graph of the relative errors as a function of the iteration number:



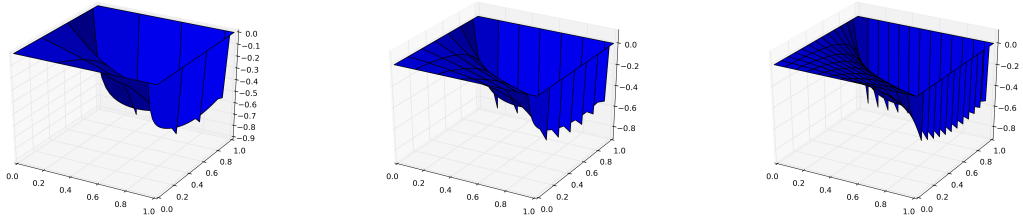
- SOR

$$u_{i,j}^{k+1} = \frac{\omega}{4}\left(u_{i-1,j}^{k} + u_{i+1,j}^{k} + u_{i,j-1}^{k} + u_{i,j+1}^{k} - h^2 f_{i,j}\right) + (1-\omega)u_{i,j}^{k}$$

where ω is the optimal ω for convergence, i.e.

$$\omega = \omega^* = 2(1-\pi h).$$

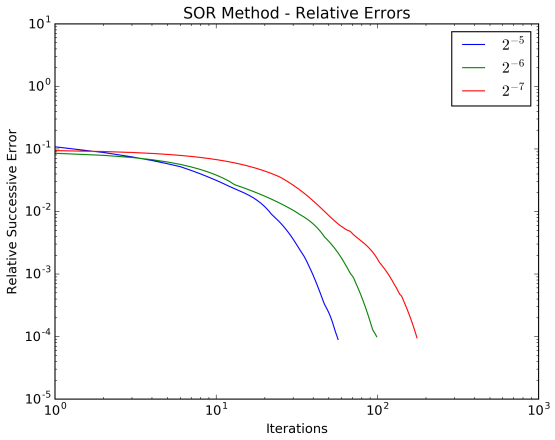
Here are the results for $h = 2^{-i}$ for $i = 5, 6, 7$.



The number of iterations required to have a relative error less than 0.0001 (that is, $\left\|u^{k+1} - u^{k}\right\|_1 < \varepsilon \|u_k\|_1$, where $\varepsilon = 0.0001$) is

h	iterations	multiplicative factor	time taken (in seconds)
2^{-5}	58		0.181905
2^{-6}	100	1.72	1.148622
2^{-7}	177	1.77	8.517767

Here is a graph of the relative errors as a function of the iteration number:



Problem 2

When solving parabolic equations numerically, one frequently needs to solve an equation of the form

$$u - \delta \nabla^2 u = f,$$

where $\delta > 0$. The analysis and numerical methods we have discussed for the Poisson equation can be applied to the above equation. Suppose we are solving the above equation on the unit square with Dirichlet boundary conditions. Use the standard five point stencil for the discrete Laplacian.

(a) Analytically compute the eigenvalues of the Jacobi iteration matrix, and show that the Jacobi iteration converges.

(b) If $h = 10^{-2}$ and $\delta = 10^{-4}$, how many iterations of SOR would it take reduce the error by a factor of 10^{-6}? How many iterations would it take for the Poisson equation? Use that the spectral radius of SOR is

$$\rho_{\text{SOR}} = \omega_{\text{opt}} - 1,$$

where

$$\omega_{\text{opt}} = \frac{2}{1 + \sqrt{1 - \rho_J^2}},$$

and where ρ_J is the spectral radius of Jacobi.

(a)

$$\begin{aligned} u - \delta \nabla^2 u &= f \\ (I - \delta A) u &= f \end{aligned}$$

but $I - \delta A = D - L - U$, so $L + U = D - I + \delta A$, thus

$$\begin{aligned} (I - \delta A) u &= f \\ (D - L - U) u &= f \\ Du &= (L + U) u + f \\ u &= (I - D^{-1}(I - \delta A)) u + D^{-1} f \\ &= Tu + C \end{aligned}$$

where

$$T = (I - D^{-1}(I - \delta A)) \qquad \text{and} \qquad C = D^{-1} f$$

We know the eigenvalues of A are

$$\mu_{ij} = \frac{2}{h^2}\cos(\ell\pi h) + \cos\big(j\pi h\big) - 2,$$

so the eigenvalues of $I - \delta A$ are $1 - \delta\mu_{ij}$, and the eigenvalues of $I - D^{-1}(I - \delta A)$ are

$$\tilde{\mu}_{ij} = 1 - \frac{1}{1 + \dfrac{4\delta}{h^2}}\big(1 - \delta\mu_{ij}\big)$$

since

$$D = \begin{bmatrix} 1 + \dfrac{4\delta}{h^2} & 0 & \dots & 0 \\ 0 & 1 + \dfrac{4\delta}{h^2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 + \dfrac{4\delta}{h^2} \end{bmatrix}$$

Thus, after simplification,

$$\tilde{\mu}_{ij} = \frac{2\delta}{h^2 + 4\delta}\big(\cos(\ell\pi h) + \cos\big(j\pi h\big)\big)$$

To prove convergence, the largest eigenvalue is found by setting $\ell = j = 1$, and thus

$$\tilde{\mu}_{ij} = \frac{4\delta}{h^2 + 4\delta}(\cos(\pi h)) < 1.$$

Since the largest eigenvalue is less than one, the iterative scheme must converge.

(b) The spectral radius ρ_J is given by

$$\rho_J = \max_{i,j} \tilde{\mu}_{ij} = \frac{4\delta}{h^2 + 4\delta}\cos(\pi h)$$

For $\delta = 10^{-4}$ and $h = 10^{-2}$, we have

$$\rho_J \approx 0.7996$$

which gives

$$\omega_{\text{opt}} = \frac{2}{1 + \sqrt{1 - \rho_J^2}} \approx 1.2496$$

and thus

$$\rho_{\text{SOR}} \approx 0.2496.$$

So for SOR, to guarantee the error is reduced by a factor of 10^{-6}, set the number of iterations k such that

$$\rho_{\text{SOR}}^k = 10^{-6}$$
$$\implies k = \frac{\log\big(10^{-6}\big)}{\log\big(\rho_{\text{SOR}}\big)} \approx 10,$$

that is, we need 10 iterations to guarantee a 6th order reduction in error. For the Poisson equation, set $\rho_J = \cos(\pi h)$ where $h = 10^{-2}$, thus $\omega_{\text{opt}} \approx 1.9391$ and so $\rho_{\text{Poisson}} \approx 0.9391$. Then

$$k = \frac{\log\big(10^{-6}\big)}{\log\big(\rho_{\text{Poisson}}\big)} \approx 220,$$

that is, we need 220 iterations to guarantee a 6th order reduction in error.

Problem 3

In this problem we compare the speed of SOR to a direct solve using Gaussian elimination. At the end of this assignment is MATLAB code to form the matrix for the 2D discrete Laplacian. The code for the 3D matrix is similar. Note that with 1 GB of memory, you can handle grids up to about 1000×1000 in 2D and $40 \times 40 \times 40$ in 3D with a direct solve. The range of grids you will explore depends on the amount of memoery you have.

(a) Solve the PDE from problem 1 using a direct solve. Put timing commands in your code and report the time to solve for a range of mesh spacings. Use SOR to solve on the same meshes and report the time and number of iterations. Comment on your results. Note that the timing results depend strongly on your implementation. Comment on the efficiency of your program.

(b) Repeat the previous part in three spatial dimensions for a range of mesh spacings. Change the right side of the equation to be a three dimensional Gaussian. Comment on your results.

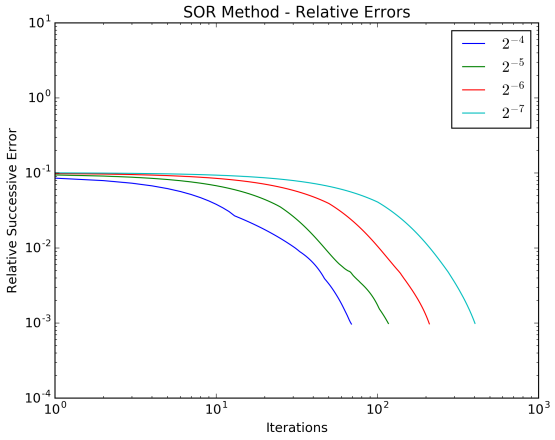
(a) Using the penta-diagonal matrix for 2D Laplacian with Dirichlet boundary conditions,

$$A = \begin{bmatrix} 0 & 0 & \dots & 0 & 1 & 0\dots & 0 & 1 & -4 & 1 & 0 & \dots & 0 & 1 & 0 & \dots & 0 & 0 \end{bmatrix},$$

where there are 1s on the first and Nth super- and sub-diagonals, and -4 on the diagonal, here are the results of using a direct solve method (python's sparse matrix solver) vs. my implementation of the SOR method (tolerance= 0.001):

h	SOR iterations	SOR iteration mult. factor	SOR time taken (in sec.)	Direct solve time taken (in sec.)
2^{-6}	63		1.091	0.01796
2^{-7}	117	1.857	6.418	0.08653
2^{-8}	210	1.795	46.421	0.55438
2^{-9}	403	1.919	363.982	3.69306

It is abundantly clear my SOR solver is not implemented to its fullest potential. Here is the graph showing the convergence in relative error for the SOR method:



(b) I chose the right hand side to be

$$f(x,y,z) = -e^{-(x-0.25)^2-(y-0.6)^2-(z-0.5)^2} \tag{0.1}$$

Using the septa-diagonal matrix for 3D Laplacian with Dirichlet boundary conditions,

$$A = \begin{bmatrix} 0 & \dots & 0 & 1 & 0 & \dots & 0 & 1 & 0 & \dots & 0 & 1 & -6 & 1 & 0 & \dots & 0 & 1 & 0 & \dots & 0 & 1 & 0 & \dots & 0 \end{bmatrix} \tag{0.2}$$

where there are 1s on the first, Nth, and N^2th super- and sub-diagonals, and -6 on the diagonal, here are the results of using a direct solve method (python's sparse matrix solver):

h	Direct solve time taken (in sec.)
2^{-6}	0.40655
2^{-7}	6.34209
2^{-8}	unable to allocate enough memory
2^{-9}	unable to allocate enough memory

I was unable to implement the SOR method in 3D.

Problem 4

Periodic boundary conditions for the on dimensional Poisson equation on $(0,1)$ are $u(0) = u(1)$ and $u_x(0) = u_x(1)$. These boundary conditions are eqsy to discretize, but lead to a singular system to solve. For example, using the standard discretization, $x_j = jh$ where $h = 1/N+1$, the discrete Laplacian at x_0 is $h^2(u_N - 2u_0 + u_1)$.

(a) Write the discrete Laplacian for periodic boundary conditions in one dimension as a matrix. Show that this matrix is singular, and find the vectors that span the null space. (Note that this matrix is symmetric, and so you have found the null space of the adjoint).

(b) What is the discrete solvability condition for the discretized Poisson equation with periodic boundary conditions in one dimension? What is the discrete solvability condition in two dimensions?

(c) Show that v is in the null space of the matrix A if and only if v is an eigenvector of the iteration matrix $T = M^{-1}N$ with eigenvalue 1, where $A = M - N$. The iteration will converge if the discrete solvability condition is satisfied provided the other eigenvalues are less than 1 in magnitude (true for Gauss-Seidel and SOR, but not for Jacobi).

(a) The discrete Laplacian for periodic boundary conditions in one dimension is given by

$$A = \begin{bmatrix} -2 & 1 & 0 & 0 & \dots & 0 & 1 \\ 1 & -2 & 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & -2 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & -2 & 1 & 0 \\ 0 & 0 & \dots & 0 & 1 & -2 & 1 \\ 1 & 0 & \dots & 0 & 0 & 1 & -2 \end{bmatrix}$$

This matrix is singular since

$$\begin{bmatrix} -2 & 1 & 0 & 0 & \dots & 0 & 1 \\ 1 & -2 & 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & -2 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & -2 & 1 & 0 \\ 0 & 0 & \dots & 0 & 1 & -2 & 1 \\ 1 & 0 & \dots & 0 & 0 & 1 & -2 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

in other words, the sum of every row is equal to 0, thus the vector of ones is in $\ker(A)$. In addition, the reduced row echelon form of A is

$$\text{rref}(A) = \begin{bmatrix} I & Q \\ \vec{0} & 0 \end{bmatrix}$$

where I is the $(N-1)\times(N-1)$ identity matrix, Q is an $(N-1)\times 1$ array of -1's and $\vec{0}$ is a $1\times(N-1)$ array of 0s. This has $N-1$ pivots, which proves $\text{rank}(A) = N-1$, and thus, by the fundamental theorem of linear algebra, $\dim(\ker(A)) = 1$, and thus $\ker(A) = \text{span}(v)$ where

$$v = \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ 1 \end{bmatrix}.$$

(b) Since A is self-adjoint, the discrete solvability condition is that $u \perp v$, that is, $\langle u, v\rangle = 0$, that is,

$$\sum_{i=1}^{N} u_i v_i = 0.$$

In two-D, the discrete Laplacian with periodic boundary conditions is

$$A = \begin{bmatrix} X & Y \\ Y^T & X \end{bmatrix}$$

where X is a tridiagonal matrix with -4 along the main diagonal and 1s along the sub- and super-diagonals, and a 1 in the bottom left corner,

$$X = \begin{bmatrix} -4 & 1 & 0 & \dots & 0 \\ 1 & -4 & 1 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \dots & 1 & -4 & 1 \\ 1 & \dots & 0 & 1 & -4 \end{bmatrix}$$

and Y has 1s along the diagonal and super-diagonal, and 1s in the top-right and bottom-left corners

$$Y = \begin{bmatrix} 1 & 1 & 0 & \dots & 1 \\ 0 & 1 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 1 & 1 & 0 \\ 1 & \dots & 0 & 1 & 1 \end{bmatrix}$$

Then the reduced row echelon form of A has the same form as the 1D equation, resulting in the same exact discrete solvability condition:

$$\sum_{i=1}^{N} u_i v_i = 0,$$

where v spans the $\ker(A)$, i.e. $v = \begin{bmatrix} 1 & 1 & \dots & 1 & 1 \end{bmatrix}^T$.

(c) v is an eigenvector of $T = M^{-1}N$ with eigenvalue 1 $\iff$ $Tv = v$ $\iff$ $M^{-1}Nv = v$ $\iff$ $Nv = Mv$ $\iff$ $0 = (M-N)v$ $\iff$ $0 = Av$ $\iff$ $v \in \text{null}(A)$. □