AMATH 585 Assignment 6

Tyler Chen

Problem 1

On the course web page is a finite difference code (steady2d.m) to solve the boundary value problem:

$$\frac{\partial}{\partial x}\left(a(x,y)\frac{\partial u}{\partial x}\right)+\frac{\partial}{\partial y}\left(a(x,y)\frac{\partial u}{\partial y}\right)=f(x,y) \quad \text{in } (0,1)\times(0,1)$$

$$u(x,0)=u(x,1)=u(0,y)=u(1,y)=0,$$

where $a(x,y)=1+x^2+y^2$ and $f(x,y)=1$. It uses a direct solver for the linear system.

Replace this direct solver first by the Jacobi method, then by the Gauss Seidel method, and then by the SOR method. For each method, make a plot of the relative residual norm, $\|b-Au^k\|/\|b\|$ versus iteration number k. (Use a logarithmic scale for the residual; i.e., you may use `semilogy` in Matlab to do the plot.) Try several different values for the parameter ω in SOR, until you find one that seems to work well.

Then try solving the linear system using the conjugate gradient method. You may write your own CG code or use the one in Matlab (called **pcg**). First try CG without a preconditioner (i.e., with preconditioner equal to the identity) and then try CG with the Incomplete Cholesky decomposition as the preconditioner. You may use `ichol` in Matlab to generate the incomplete Cholesky decomposition. Again make a plot of relative residual norm versus iteration number for the CG method.

Experiment with a few different mesh sizes and comment on how the number of iterations required to reach a fixed level of accuracy seems to vary with h for each method.

Solution

We implement Jacobi iteration, Gauss-Seidel, and successive over relaxation:

```
def jacobi(A,b,tol,max_iter=20):
    n = np.shape(A)[1]
    x = sparse.csc_matrix((n,1))

    M = sparse.diags(A.diagonal(), format='csc')

    residual_norm=np.zeros(max_iter)
    iter_tol = -1
    for iter in range(max_iter):
        residual = b-A.dot(x)
        x += sparse.linalg.spsolve_triangular(M,residual)
        residual_norm[iter] = np.linalg.norm(residual)/np.linalg.norm(b)
        if residual_norm[iter] < tol:
            iter_tol = iter
            break

    return (x,residual_norm,iter_tol)
```

```
def gauss_seidel(A,b,tol,max_iter=20):
    n = np.shape(A)[1]
    x = sparse.csc_matrix((n,1))

    M = sparse.tril(A,0, format='csc')

    residual_norm=np.zeros(max_iter)
    iter_tol = -1
    for iter in range(max_iter):
```

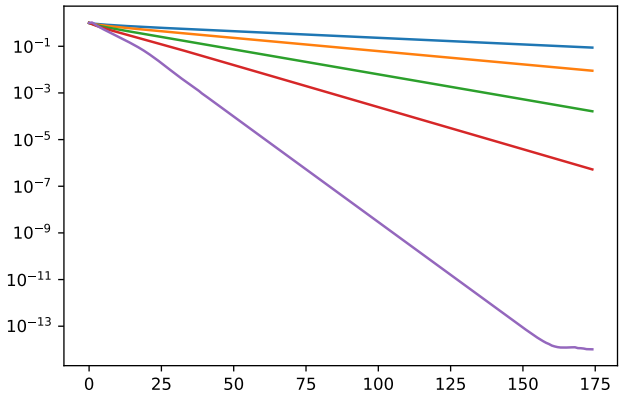
```
        residual = b-A.dot(x)
        x += sparse.linalg.spsolve_triangular(M,residual)
        residual_norm[iter] = np.linalg.norm(residual)/np.linalg.norm(b)
        if residual_norm[iter] < tol:
            iter_tol = iter
            break

    return (x,residual_norm,iter_tol)
```

```
def SOR(A,b,w,tol,max_iter=20,opt=False):
    n = np.shape(A)[1]
    x = sparse.csc_matrix((n,1))

    D = sparse.diags(A.diagonal(), format='csc')
    L = sparse.tril(A,-1, format='csc')
    M = D/w+L

    if opt:
        G = sparse.eye(n) - sparse.diags(1/A.diagonal(),format='csc').dot(A)
        p = np.linalg.norm(sparse.linalg.eigs(G,1,which='LM')[0])
        w = 2/(1+np.sqrt(1-p**2))
        print(w)
    residual_norm=np.zeros(max_iter)

    iter_tol = -1
    for iter in range(max_iter):
        residual = b-A.dot(x)
        x += sparse.linalg.spsolve_triangular(M,residual)
        residual_norm[iter] = np.linalg.norm(residual)/np.linalg.norm(b)
        if residual_norm[iter] < tol:
            iter_tol = iter
            break

    return (x,residual_norm,iter_tol)
```

Note that we use a incomplete LU decomposition to compute a preconditioner for the conjugate gradient method. This is done using Scipy's sparse package as:

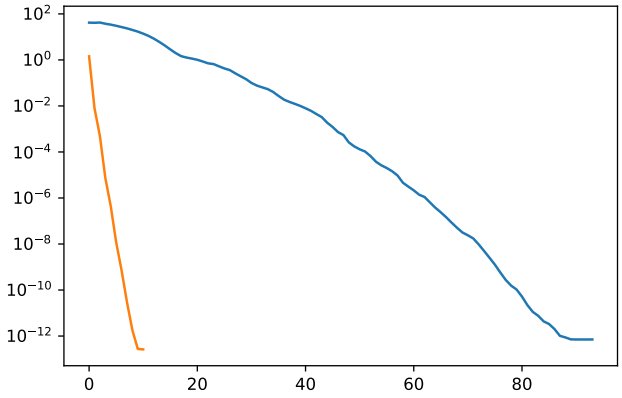
```
    lu = sparse.linalg.spilu(A,drop_tol = 1e-3)
    M = lu.solve(np.identity(N))
```

Figure 2 shows how the residual of each of these methods vs the iteration number. As expected Gauss-Seidel converges about twice as fast as Jacobi. Successive over relaxation converges faster than either of these methods, with the rate depending on the choice of ω.



blue: Jacobi
orange: GS
green: SOR ($\omega = 1.3$)
red: SOR ($\omega = 1.5$)
purple: SOR ($\omega = 1.7$)

Figure 1: Residual norm vs. iteration count with $n = 20$ for simple iterative methods

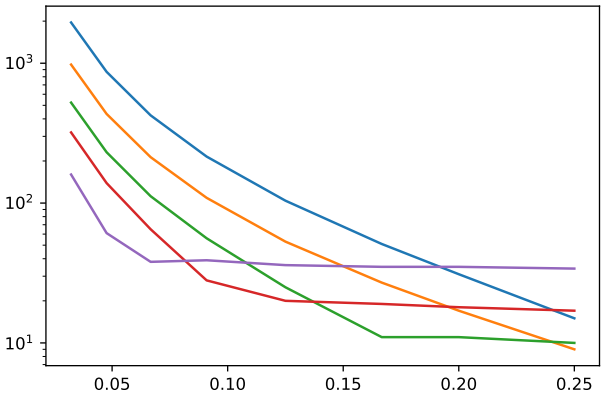


blue: no preconditioner
orange: iLU preconditioner

Figure 2: Residual norm vs. iteration count with $n = 20$ for conjugate gradient methods

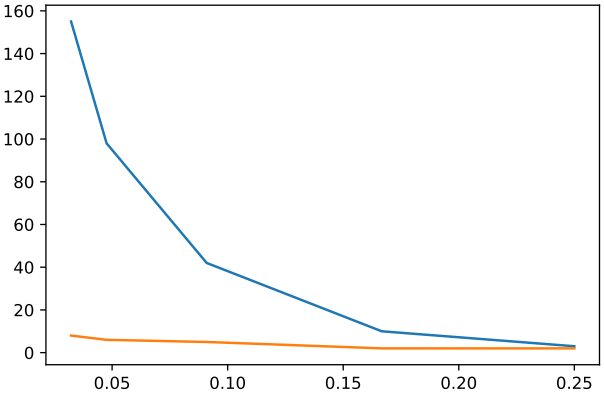
Figures 3 and 4 show the number of iterations required to reach the specified tolerance for Jacobi iteration, Gauss-Seidel iteration, successive over relaxation at varying ω, conjugate gradient, and conjugate gradient using the preconditioner described above. All methods take longer to converge as the size of the system increases. However, using a preconditioner means that the number of iterations does not increase significantly.

Jacobi and Gauss-Seidel iteration require significantly more iterations as the system size increases. Successive over relaxation does seems to remain relatively constant until some threshold size at which point the number of iterations begins to increase rapidly. This may be explained by the choice of ω. We expect that as the system size increases that the optimal value of omega converges to 2. As seen in the figure, the higher values of ω worked fine for longer.



blue: Jacobi
orange: GS
green: SOR ($\omega = 1.3$)
red: SOR ($\omega = 1.5$)
purple: SOR ($\omega = 1.7$)

Figure 3: Iterations needed (log scale) to reach tolerance of 10^{-5} vs. mesh size.



blue: no preconditioner
orange: iLU preconditioner

Figure 4: Iterations to reach tolerance of 10^{-16} vs. mesh size

Problem 2

Suppose a symmetric positive definite matrix A has one thousand eigenvalues uniformly distributed between 1 and 10, and one eigenvalue of 10^4. Suppose another symmetric positive definite matrix B has an eigenvalue of 1 and has one thousand eigenvalues uniformly distributed between 10^3 and 10^4. Since each matrix has condition number $\kappa = 10^4$, we have seen that the error at step k of the CG algorithm satisfies

$$\frac{\|e^{(k)}\|_{A,B}}{\|e^{(0)}\|_{A,B}} \leq 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^k = 2\left(\frac{99}{101}\right)^k.$$

Give another bound on $\|e^{(k)}\|_A/\|e^{(0)}\|_A$ based on a polynomial that is a product of a degree $k-1$ Tchebyshev polynomial on the interval $[1, 10]$ and a linear polynomial that is 1 at the origin and 0 at 10^4. Give another bound on $\|e^{(k)}\|_B/\|e^{(0)}\|_B$ based on a polynomial that is a product of a degree $k-1$ Tchebyshev polynomial on the interval $[10^3, 10^4]$ and a linear polynomial that is 1 at the origin and 0 at 1. For which matrix would you expect CG to converge more rapidly? [If you are not sure, you may try it in Matlab.]

Solution

Let $P_k(x)$ be the polynomial constructed implicitly by the conjugate gradient method for a (reasonable) matrix X. That is,

$$P_k = \text{argmin}_{P\in\mathcal{P}_k} \|P(A)e_0\|_X$$

Recall that for any polynomial, $P(x)$ we have,

$$\frac{\left\|e^{(k)}\right\|_X}{\left\|e^{(0)}\right\|_X} = \frac{\|P_k(A)e_0\|_X}{\|e_0\|_X} \leq \frac{\|P(A)e_0\|_X}{\|e_0\|_X} \leq \max_{1\leq j\leq m} |P(\lambda_j)|$$

Recall further that,

$$\left|\frac{T_k(1)}{T_k\left(\frac{\lambda_b+\lambda_a}{\lambda_b-\lambda_a}\right)}\right| \leq 2\left(\frac{\sqrt{\kappa'}-1}{\sqrt{\kappa'}+1}\right)^{k-1}, \qquad \kappa' = \frac{\lambda_b}{\lambda_a}$$

Let $\{\lambda_j\}_{j=1}^m$ be the eigenvalues of A in order. That is $\lambda_1 = 1, \lambda_{m-1} = 10, \lambda_m = 10^4$. Define,

$$\tilde{P}_{A,k}(x) = \frac{T_{k-1}\left(\frac{\lambda_{m-1}+\lambda_1-2x}{\lambda_{m-1}-\lambda_1}\right)}{T_{k-1}\left(\frac{\lambda_{m-1}+\lambda_1}{\lambda_{m-1}-\lambda_1}\right)}\left(\frac{\lambda_m - x}{\lambda_m}\right)$$

Clearly $\tilde{P}_{A,k}(\lambda_m) = 0$. We therefore know $\max_j \tilde{P}_{A,k}(\lambda_j)$ is attained for some $\lambda_j \in$, $j = 1, \ldots, m-1$. Since T_k attains its maximum on $[0, 1]$ at the boundaries, and since $|\lambda_m - x|$ is strictly decreasing on $[\lambda_1, \lambda_{m-1}]$, we know,

$$\max_j |\tilde{P}_{A,k}(\lambda_j)| = |\tilde{P}_{A,k}(\lambda_1)|$$

Thus, with $\kappa_A = \lambda_{m-1}/\lambda_1 = 10$ and $|\lambda_m - \lambda_1|/|\lambda_m| = (10^4 - 1)/10^4 \leq 1$,

$$\frac{\left\|e^{(k)}\right\|_A}{\left\|e^{(0)}\right\|_A} \leq 2\left(\frac{\sqrt{\kappa_A}-1}{\sqrt{\kappa_A}+1}\right)^{k-1}\left|\frac{\lambda_m-\lambda_1}{\lambda_m}\right| \leq 2\left(\frac{\sqrt{10}-1}{\sqrt{10}+1}\right)^{k-1}$$

Now, let $\{\gamma_j\}_{j=1}^m$ be the eigenvalues of B in order. That is $\gamma_1 = 1, \gamma_2 = 10^3, \gamma_m = 10^4$. Define,

$$\tilde{P}_{B,k}(x) = \frac{T_{k-1}\left(\frac{\gamma_{m-1}+\gamma_1-2x}{\gamma_{m-1}-\gamma_1}\right)}{T_{k-1}\left(\frac{\gamma_{m-1}+\gamma_1}{\gamma_{m-1}-\gamma_1}\right)}\left(\frac{\gamma_1 - x}{\gamma_1}\right)$$

Clearly $\tilde{P}_{A,k}(\gamma_1) = 0$. We therefore know $\max_j \tilde{P}_{A,k}(\gamma_j)$ is attained for some $\gamma_j \in$, $j = 1, \ldots, m-1$. Since T_k attains its maximum on $[0,1]$ at the boundaries, and since $|\gamma_1 - x|$ is strictly increasing on $[\gamma_2, \gamma_m]$, we know,

$$\max_j |\tilde{P}_{A,k}(\gamma_j)| = |\tilde{P}_{A,k}(\gamma_m)|$$

Thus, with $\kappa_B = \gamma_m/\gamma_2 = 10$ and $|\gamma_1 - \gamma_m|/|\gamma_1| = (10^4 - 1)/1 \leq 10^4$,

$$\frac{\left\|e^{(k)}\right\|_B}{\left\|e^{(0)}\right\|_B} \leq 2\left(\frac{\sqrt{\kappa_B}-1}{\sqrt{\kappa_B}+1}\right)^{k-1}\left|\frac{\gamma_1-\gamma_m}{\gamma_1}\right| \leq 20000\left(\frac{\sqrt{10}-1}{\sqrt{10}+1}\right)^{k-1}$$

Presumably conjugate gradient applied to A will converge faster as the bound on convergence is much better. However we have not proved this here as our bounds are only upper bounds.

Problem 3

Repeat the experiments on p. 103 of the text, leading to Figures 4.8 and4.9, but use the Gauss-Seidel method and (unpreconditioned) CG instead of Jacobi iteration. That is, set up difference equations for the problem

$$u''(x) = f(x), \quad u(0) = 1,\ u(1) = 3,$$

where

$$f(x) = -20 + a\phi''(x)\cos(\phi(x)) - a(\phi'(x))^2 \sin(\phi(x)),$$

where $a = 0.5$ and $\phi(x) = 20\pi x^3$. The true solution is

$$u(x) = 1 + 12x - 10x^2 + a\sin(\phi(x)).$$

Starting with initial guess $u^{(0)}$ with components $1 + 2x_i$, $i = 1, \ldots, 255$, run, say, 20 Gauss-Seidel iterations and then 20 CG iterations, plotting the true solution to the linear system and the approximate solution, say, at steps 0, 5, 10, and 20, and also plotting the error (the difference between true and approximate solution). Print the size of the error (the L_2-norm or the ∞-norm) at these steps too. Based on your results, would you say that Gauss-Seidel and CG would be effective smoothers for a multigrid method?

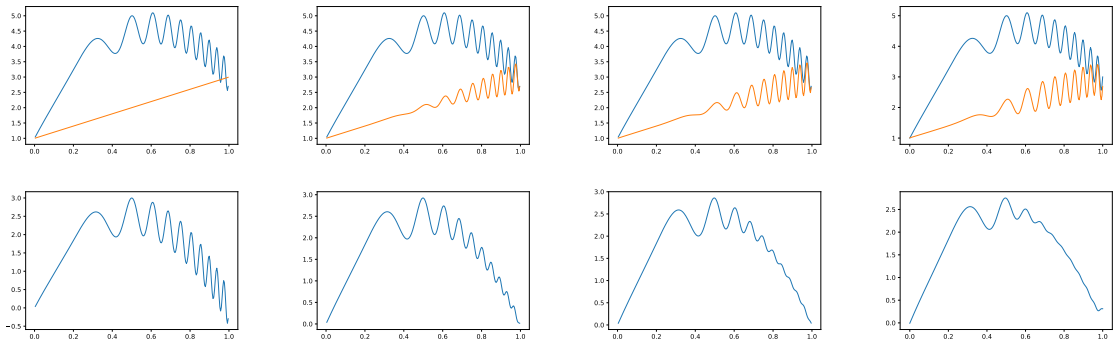
Solution

We would like to solve $Au = b$ where, A has $-2/h^2$ on the main diagonal and $1/h^2$ on the off diagonals, and b is f evaluated on the grid where we subtract $1/h^2$ from the first entry of b and $3/h^2$ from the last entry to take care of the boundary conditions.

To ensure A is positive definite we multiply both sides by -1.

Figures 5 and 6 show the approximate solution (orange) and the actual solution (blue) on the top figures. The bottom figure is the difference of the two.

Both seem like they would work fine as smoothers for multigrid methods as high frequency parts of the error are reduced quickly.



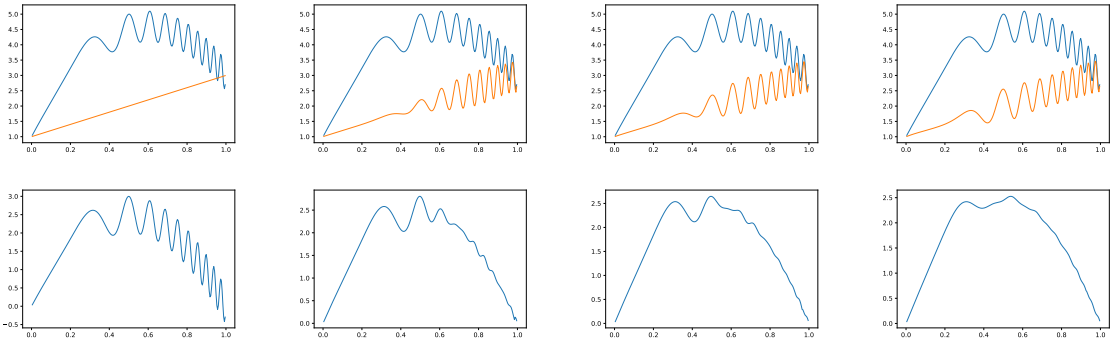
$\|\text{error}\|_{L_2} = 1.9061$ (a) $k = 0$

$\|\text{error}\|_{L_2} = 1.8919$ (b) $k = 5$

$\|\text{error}\|_{L_2} = 1.8870$ (c) $k = 10$

$\|\text{error}\|_{L_2} = 1.8818$ (d) $k = 20$

Figure 5: Gauss Seidel Method



$\|\text{error}\|_{L_2} = 1.9061$ $\|\text{error}\|_{L_2} = 1.8845$ $\|\text{error}\|_{L_2} = 1.8776$ $\|\text{error}\|_{L_2} = 1.8610$

(a) $k = 0$ (b) $k = 5$ (c) $k = 10$ (d) $k = 20$

Figure 6: Conjugate Gradient Method