Homework 6

Due: March 10, 2021

1. The conjugate gradient method for solving a symmetric positive definite linear system $Ax = b$ can be written as below:

```
Given $x_0$, compute $r_0 = b - Ax_0$, and set $p_0 = r_0$.
For $k = 1, 2, \ldots$,
  Compute $Ap_{k-1}$.
  Set $x_k = x_{k-1} + a_{k-1}p_{k-1}$, where $a_{k-1} = \frac{\langle r_{k-1}, r_{k-1}\rangle}{\langle p_{k-1}, Ap_{k-1}\rangle}$.
  Compute $r_k = r_{k-1} - a_{k-1}Ap_{k-1}$.
  Set $p_k = r_k + b_{k-1}p_{k-1}$, where $b_{k-1} = \frac{\langle r_k, r_k\rangle}{\langle r_{k-1}, r_{k-1}\rangle}$.
Endfor
```

(a) Show that the residual vectors $r_0, \ldots, r_k$ are orthogonal to each other ($\langle r_i, r_j\rangle = 0$ if $i \neq j$) and that the direction vectors $p_0, \ldots, p_k$ are A-orthogonal ($\langle p_i, Ap_j\rangle = 0$ if $i \neq j$). [Hint: First show that $\langle r_1, r_0\rangle = \langle p_1, Ap_0\rangle = 0$ and then use induction on k.]

(b) If A is the $N \times N$ matrix of the 5-point operator for Poisson's equation on a square, count the number of operations (additions, subtractions, multiplications, and divisions) performed in each iteration. (Show how you arrive at your result.)

(c) Compare your operation count in (b) to that of a Gauss-Seidel iteration applied to the same N by N 5-point matrix A. Also compare to the operation count for a multigrid V-cycle, using one Gauss-Seidel iteration on each visit to each grid level. (Again, show how you derive your operation counts.)

Solution.

(a) We have

$$
\begin{aligned}
\langle r_0, r_1\rangle &= \langle r_0, r_0 - a_0 Ap_0\rangle \\
&= \langle r_0, r_0 - \frac{\langle r_0, r_0\rangle}{\langle r_0, Ar_0\rangle} Ar_0\rangle \\
&= \langle r_0, r_0\rangle - \frac{\langle r_0, r_0\rangle}{\langle r_0, Ar_0\rangle}\langle r_0, Ar_0\rangle \\
&= 0
\end{aligned}
$$

Similarly

$$\begin{aligned}
\langle p_1, Ap_0\rangle &= \langle r_1 + b_0 p_0, Ap_0\rangle \\
&= \langle r_1 + \frac{\langle r_1, r_1\rangle}{\langle r_0, r_0\rangle} r_0, Ar_0\rangle \\
&= \langle r_1, Ar_0\rangle + \frac{\langle r_1, r_1\rangle}{\langle r_0, r_0\rangle}\langle r_0, Ar_0\rangle \\
&= \langle r_1, \frac{r_0 - r_1}{a_0}\rangle + \frac{\langle r_1, r_1\rangle}{a_0} \\
&= \frac{\langle r_1, r_0\rangle}{a_0} - \frac{\langle r_1, r_1\rangle}{a_0} + \frac{\langle r_1, r_1\rangle}{a_0} \\
&= 0
\end{aligned}$$

This proves $\langle r_i, r_j\rangle = \langle p_i, Ap_j\rangle = 0$ for $i = 1$ and $j < 1$, which is our base case. Assume $\langle r_i, r_j\rangle = \langle p_i, Ap_j\rangle = 0$ holds for $i = k$ and $j < k$. Then

$$\begin{aligned}
\langle p_k, Ap_j\rangle &= \langle Ap_k, p_j\rangle \\
&= \langle Ap_k, r_j + b_{j-1}p_{j-1}\rangle \\
&= \langle Ap_k, r_j\rangle + \langle Ap_k, b_{j-1}p_{j-1}\rangle \\
&= \langle Ap_k, r_j\rangle + 0 = 0
\end{aligned}$$

Then $\langle Ap_k, r_j\rangle = 0$. Consider the case $i = k + 1$

$$\begin{aligned}
\langle r_{k+1}, r_j\rangle &= \langle r_k - a_k Ap_k, r_j\rangle \\
&= \langle r_k, r_j\rangle - a_k\langle Ap_k, r_j\rangle \\
&= 0 \\
\langle p_{k+1}, Ap_j\rangle &= \langle r_{k+1} + b_k p_k, Ap_j\rangle \\
&= \langle r_{k+1}, Ap_j\rangle + b_k\langle p_k, Ap_j\rangle \\
&= \langle r_{k+1}, \frac{r_j - r_{j+1}}{a_j}\rangle \\
&= \frac{\langle r_{k+1}, r_j\rangle}{a_j} - \frac{\langle r_{k+1}, r_{j+1}\rangle}{a_j} \\
&= 0
\end{aligned}$$

For $j = k$

$$\begin{aligned}
\langle r_{k+1}, r_k\rangle &= \langle r_k - a_k A p_k, r_k\rangle \\
&= \langle r_k, r_k\rangle - a_k\langle A p_k, r_k\rangle \\
&= \langle r_k, r_k\rangle - \langle r_k, r_k\rangle \\
&= 0 \\
\langle p_{k+1}, A p_k\rangle &= \langle r_{k+1} + b_k p_k, A p_k\rangle \\
&= \langle r_{k+1}, A p_k\rangle + b_k\langle p_k, A p_k\rangle \\
&= \langle r_{k+1}, \frac{r_k - r_{k+1}}{a_k}\rangle + \frac{\langle r_{k+1}, r_{k+1}\rangle}{\langle r_k, r_k\rangle}\frac{\langle r_k, r_k\rangle}{a_k} \\
&= \frac{\langle r_{k+1}, r_k\rangle}{a_k} \\
&= 0
\end{aligned}$$

The induction follows, thus $\langle r_i, r_j\rangle = \langle p_i, A p_j\rangle = 0$ holds for all $i \neq j$.

(b) If A is $N \times N$, then x_k, p_k, r_k are $N \times 1$. Since A is a 5-point operator, it has at most 5 nonzero values in any row. Multiplying A with a vector will take 5 multiplications and 4 additions for each row, so in total $9N$ operations. We count the number of operations in each step of the algorithm

(1) 1 matrix multiplication ($9N$).

(2) 2 inner products ($2N - 1$ for each), 1 scalar division (assuming $N \gg 1$ this is negligible), 1 scalar multiplication (N), 1 vector addition (N). In total $6N$.

(3) 1 scalar-vector multiplication (N), 1 vector subtraction (N). In total $2N$.

(4) 1 inner product ($2N - 1$), 1 scalar multiplication (N), 1 vector addition (N). In total $4N$.

Adding these together we have $O(21N)$ operations in each iteration.

(c) We count each step in Gauss-Seidel

(1) 1 matrix multiplication ($9N$), 1 vector subtraction (N). In total $10N$.

(2) Since M is lower triangular, there are at most 3 nonzero values in any row. We use back substitution to solve $Mz_k = r_k$, i.e. $az_i + bz_j + cz_k = r_k$. z_i and z_j are computed in previous iterations, thus solving for z_k requires 2 multiplications, 2 subtractions and 1 division. We do this for N rows, totalling $5N$ operations.

(3) 1 vector addition (N).

Adding these together we have $O(16N)$ operations for Gauss-Seidel.

For a multigrid V-cycle, suppose we have J grid levels. We do one Gauss-Seidel iteration on each visit to each grid level, in total two iterations at each level. For each successive level the number of operations is reduced by a factor of 4, since the number of grid points in each dimension is reduced by a factor of 2. Then the total number of iterations is

$$2 \cdot O(16N) \sum_{j=0}^{J} (\frac{1}{4})^j$$

Taking $J \to \infty$ we get $O(\frac{128}{3}N)$. We are aware that there are other steps in a multigrid V-cycle, such as projection to a lower grid, interpolating back to an upper grid and solving for the error at the lowest grid. We think these require small operation counts compared to Gauss-Seidel at each level, thus we do not take them into consideration.

2. Repeat the experiments on p. 103 of the text, leading to Figures 4.8 and 4.9, 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 set up the A matrix as 2.10 in LeVeque and the f vector with boundary terms subtracted from f_1 and f_m. Note when we use `pcg` in MATLAB, we need to make sure A is positive definite, thus we multiply everything by -1. Based on our results, both methods would be effective smoother for the multigrid method since oscillatory parts of the error are reduced quickly with the iteration number.

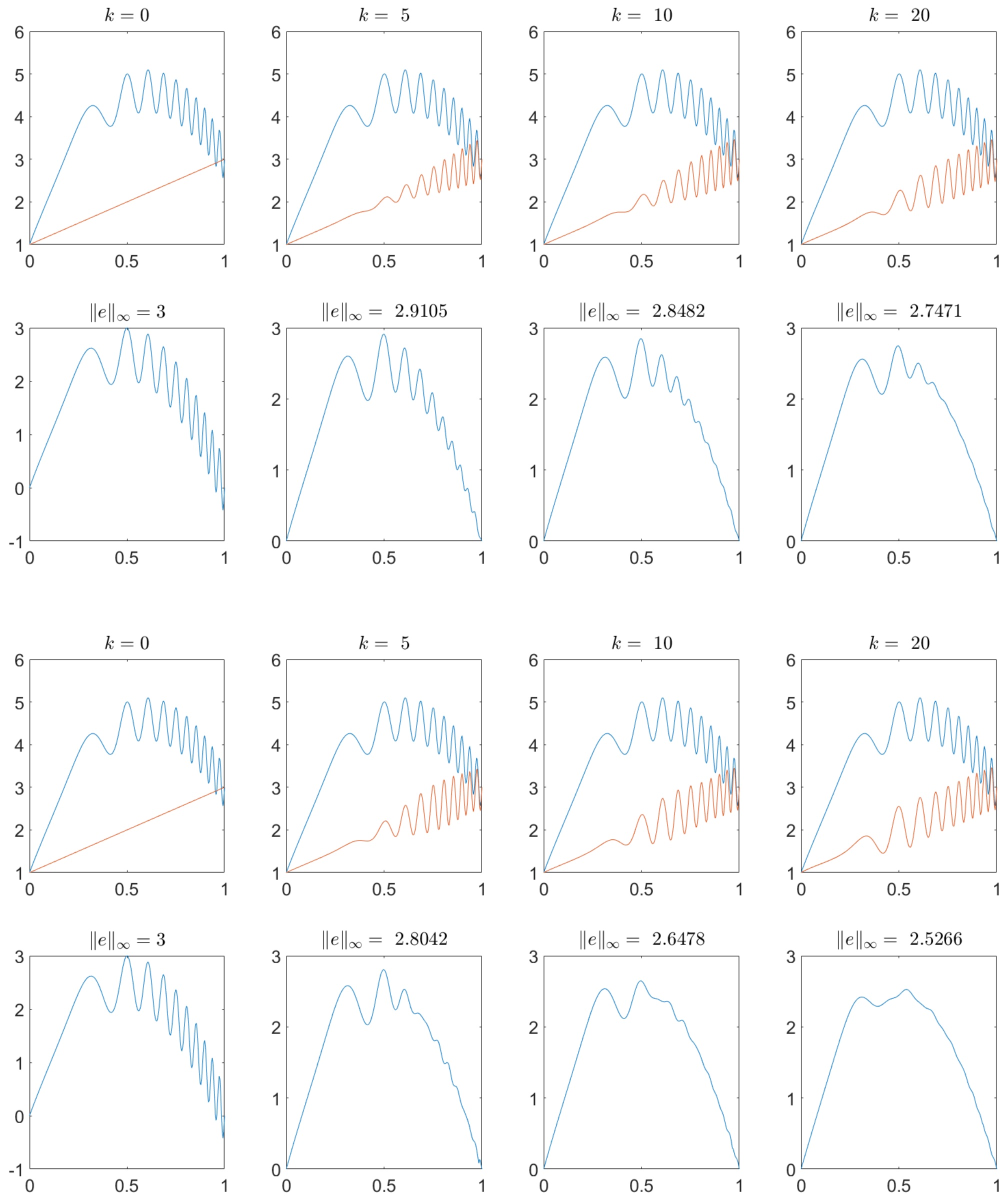


Figure 1: Gauss-Seidel (top) and CG methods (bottom).

3. Implement a 2-grid method for solving the 1D model problem with homogeneous Dirichlet boundary conditions:

$$u_{xx} = f(x), \quad u(0) = u(1) = 0.$$

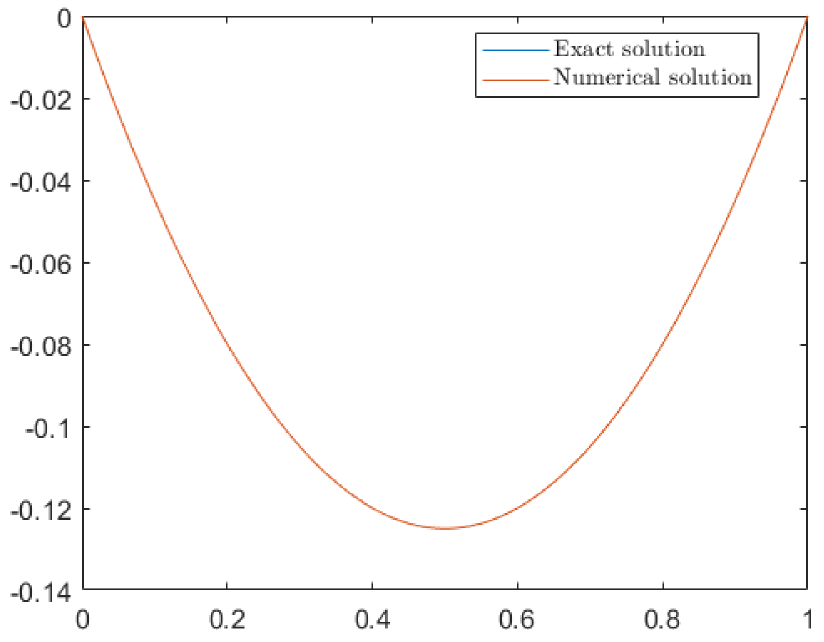
Use linear interpolation to go from the coarse grid with spacing $2h$ to the fine grid with spacing h. Take the projection matrix I_h^{2h}, going from the fine grid to the coarse grid, to be 0.5 times the transpose of the interpolation matrix: $I_h^{2h} = \frac{1}{2}(I_{2h}^h)^T$. Use a multigrid V-cycle with 1 smoothing step on each visit to each grid level. Try weighted Jacobi and Gauss-Seidel as the smoothing step. Try several different values of the mesh spacing h and show that you achieve convergence to a fixed tolerance in a number of cycles that is independent of the mesh size.

Solution.

We implement a 2-grid method in MATLAB

```
function [u,iter]=twogrid(u,A,f,tol)
m=length(A);
M=3/2*sparse(diag(diag(A))); % weighted jacobi
% M=sparse(tril(A)); % gauss-seidel
mc=(m-1)/2; % assume m is odd
h=1/(m+1);
Ac=(-2*diag(ones(1,mc))+diag(ones(1,mc-1),1)+diag(ones(1,mc-1),-1))/h^2;
I_2hh=zeros(m,mc); % interpolation matrixeven
block=[1/2 1 1/2]';
for i=1:mc
    I_2hh(2*i-1:2*i+1,i)=block;
end
I_h2h=0.5*I_2hh';% projection matrix
% 2-grid
r=f-A*u;
r_norm=1;
iter=0;
while r_norm>tol
    rc=I_h2h*r; % projection
    zc=Ac\rc; % solve in coarse grid
    z=I_2hh*zc; % interpolate
    u=u+z;
    r=f-A*u;
    u=u+M\r; % relaxation
    r=f-A*u;
    r_norm=norm(r)/norm(f);
    iter=iter+1;
end
```

Consider $f(x) = 1$, the exact solution is $u = \frac{1}{2}(x^2 - x)$. We solve this with weighted Jacobi ($\omega = \frac{2}{3}$, as suggested in Wikipedia) and Gauss-Seidel relaxation steps.



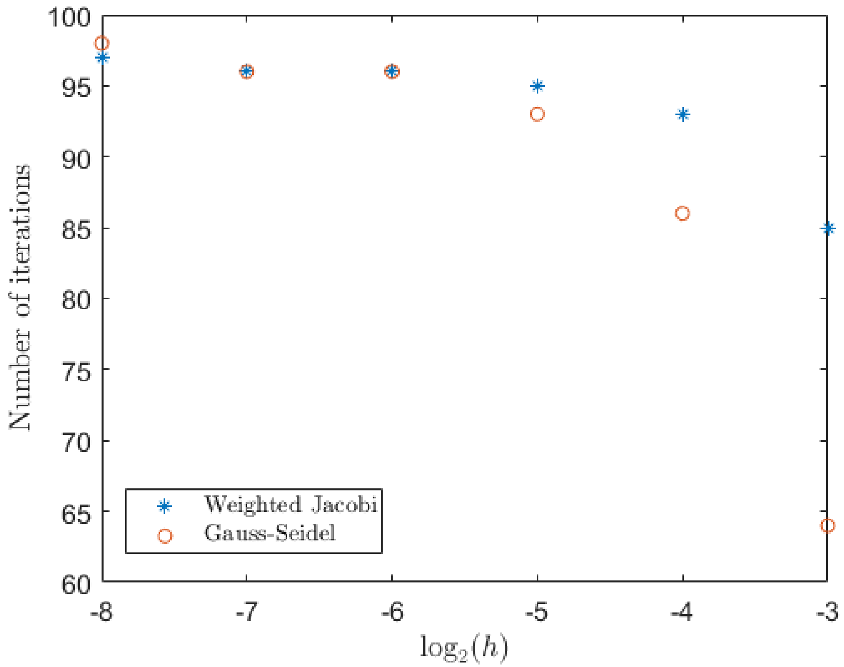


Figure 2: Top: two-grid method with weighted Jacobi relaxation. Bottom: Number of iterations required for residual smaller than 10^{-12} against mesh size. We observe that the number of iterations is independent of mesh size.

4. (a) Consider an iteration of the form

$$x_k = x_{k-1} + M^{-1}(b - Ax_{k-1}),$$

for solving a nonsingular linear system $Ax = b$. Note that the error $e_k := A^{-1}b - x_k$ satisfies

$$e_k = (I - M^{-1}A)e_{k-1} = \ldots = (I - M^{-1}A)^k e_0.$$

Assume that $\|e_0\|_2 = 1$ and that $\|I - M^{-1}A\|_2 = \frac{1}{2}$. Estimate the number of iterations required to reduce the 2-norm of the error below 2^{-20}. Show how you obtain your estimate. Now suppose you know only that the spectral radius $\rho(I - M^{-1}A) = \frac{1}{2}$. Can you give an estimate of the number of iterations required to reduce the 2-norm of the error below 2^{-20}? Explain why or why not.

(b) Consider the GMRES algorithm applied to an n by n matrix A with the sparsity pattern pictured below:

$$\begin{bmatrix} * & * & \cdots & * & * \\ * & * & \cdots & * & 0 \\ 0 & * & \cdots & * & 0 \\ \vdots & \ddots & \ddots & \vdots & \vdots \\ 0 & \cdots & \cdots & * & 0 \end{bmatrix},$$

where the $*$'s represent nonzero entries. Show that if the initial residual is the nth unit vector $(0, \ldots, 0, 1)^T$, then the algorithm makes no progress until step n. Show that a matrix with this sparsity pattern can have *any* eigenvalues. Conclude that eigenvalue information alone cannot be enough to ensure fast convergence of the GMRES algorithm.

Solution.

(a) Taking the 2-norm of the relation

$$\|e_k\|_2 \le \|I - M^{-1}A\|_2^k \|e_0\|_2 = \frac{1}{2}^k$$

If $\|e_k\|_2 \le 2^{-20}$, then $k = 20$. If we only know about the spectral radius, we cannot estimate the number of iterations, because we only have

$$\rho(I - M^{-1}A) = \frac{1}{2} \le \|I - M^{-1}A\|_2$$

We need to know about the eigenvector matrix of $I - M^{-1}A$ to get an estimate.

(b) Given $r^{(0)} = (0, \ldots, 0, 1)^T$, we know $Ar^{(0)} = (*, \ldots, 0, 1)^T$, $A^2r^{(0)} = (*, *, \ldots, 0, 1)^T$ and therefore for $k < n$, $A^k r^{(0)}$ has some zero entries and is orthogonal to $r^{(0)}$. The residual is updated by

$$r^{(k)} = r^{(0)} - \text{span}\left\{Ar_0, A^2r_0, \ldots, A^k r_0\right\}$$

Since $r^{(0)}$ is orthogonal to all vectors in the span, it will stay unchanged unless $k \geq n$. Then $r^{(0)}$ will no longer be orthogonal to $A^k r^{(0)}$, thus the algorithm can reduce $r^{(0)}$ with each iteration.

Consider a monic polynomial whose roots are $\lambda_k \in \mathbb{C}$ for $k = 1, \ldots, n$

$$a(x) = a_0 + a_1 x + \ldots + a_{n-1} x^{n-1} + x^n$$

Its companion matrix is defined as

$$\text{A} = \begin{bmatrix} 0 & 0 & \cdots & 0 & -a_0 \\ 1 & 0 & \cdots & 0 & -a_1 \\ 0 & 1 & \cdots & 0 & -a_2 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & -a_{n-1} \end{bmatrix}$$

which has eigenvalues λ_k. Consider an unitary antidiagonal matrix

$$Q = \begin{bmatrix} & & & & 1 \\ & & & 1 & \\ & & \ddots & & \\ & 1 & & & \\ 1 & & & & \end{bmatrix}$$

We have

$$(QAQ^*)^* = \begin{bmatrix} -a_{n-1} & \cdots & -a_2 & -a_1 & -a_0 \\ 1 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix}$$

which is the sparsity pattern we want. Note that QAQ^* is similar to A, thus it has λ_k as eigenvalues. $(QAQ^*)^*$ is the conjugate transpose of QAQ^*, thus its eigenvalues are λ_k^*. Since λ_k can have any value, so does λ_k^*. Thus we prove that a matrix with this sparsity pattern can have any eigenvalues and GMRES will not make progress until step n.