APPM 5610 - Homework 5

1a) $A = \begin{bmatrix} a_1 & c_1 & & \\ b_2 & \ddots & \ddots & \\ & \ddots & \ddots & c_{n-1} \\ & & b_n & a_n \end{bmatrix} \in \mathbb{R}^{n \times n}$, $A_3 = \begin{bmatrix} a_1 & c_1 & 0 \\ b_2 & a_2 & c_2 \\ 0 & b_3 & a_3 \end{bmatrix}$

To determine the LU decomposition of A when it's a 3×3 tridiagonal matrix, we write $A = LU$, where

$$\underbrace{\begin{bmatrix} a_1 & c_1 & 0 \\ b_2 & a_2 & c_2 \\ 0 & b_3 & a_3 \end{bmatrix}}_{=A} = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ x_1 & 1 & 0 \\ 0 & x_2 & 1 \end{bmatrix}}_{=L} \underbrace{\begin{bmatrix} a_1 & c_1 & 0 \\ 0 & y_1 & z_1 \\ 0 & 0 & y_2 \end{bmatrix}}_{=U}$$

Multiplying L with U and matching the appropriate entries of LU with the corresponding entries in A, we get the following equations:

$$b_2 = x_1 a_1 \quad \rightarrow \quad x_1 = \frac{b_2}{a_1}$$

$$a_2 = x_1 c_1 + y_1 \quad \rightarrow \quad y_1 = a_2 - \frac{b_2}{a_1} c_1$$

$$c_2 = z_1 \quad \rightarrow \quad z_1 = c_2$$

$$b_3 = x_2 y_1 \quad \rightarrow \quad x_2 = \frac{b_3}{a_2 - \frac{b_2}{a_1} c_1}$$

$$a_3 = x_2 z_1 + y_2 \quad \rightarrow \quad y_2 = a_3 - \frac{b_3}{a_2 - \frac{b_2}{a_1} c_1} c_2$$

That is, $A = LU$, where

$$L = \begin{bmatrix} 1 & 0 & 0 \\ \frac{b_2}{a_1} & 1 & 0 \\ 0 & \frac{b_3}{a_2 - \frac{b_2}{a_1} c_1} & 1 \end{bmatrix}, \quad U = \begin{bmatrix} a_1 & c_1 & 0 \\ 0 & a_2 - \frac{b_2}{a_1} c_1 & c_2 \\ 0 & 0 & a_3 - \frac{b_3}{a_2 - \frac{b_2}{a_1} c_1} c_2 \end{bmatrix}$$

1b) For a general $n \times n$ tri-diagonal matrix A, we have that A's LU decomposition is given by

$$L_{j,j} = 1 \quad \text{for} \quad j = 1, 2, \ldots, n$$

$$L_{2,1} = \frac{b_2}{a_1}$$

$$L_{j,j-1} = \frac{b_j}{a_{j-1} - (L_{j-1,j-2}) c_{j-2}}, \quad j = 3, \ldots, n$$

L is zero everywhere else

$$u_{1,1} = a_1$$

$$u_{1,2} = c_1$$

$$u_{j,j} = a_j - (L_{j,j-1}) c_{j-1}$$

$$u_{j,j+1} = c_j$$

U is zero everywhere else

2b) We solve the system using 4 digit floating point arithmetic with rounding:

$$\left[\begin{array}{ccc|c} 6 & 2 & 2 & -2 \\ 2 & 0.6667 & 0.3333 & 1 \\ 1 & 2 & -1 & 0 \end{array}\right] \underset{\substack{R_2 = R_2 - \frac{2}{6}R_1 \\ R_3 = R_3 - \frac{1}{6}R_1}}{\sim} \left[\begin{array}{ccc|c} 6 & 2 & 2 & -2 \\ 0 & 0.0001 & -0.3333 & 1.667 \\ 0 & 1.667 & -1.333 & 0.3334 \end{array}\right]$$

$$\underset{R_3 = R_3 - 16670R_2}{\sim} \left[\begin{array}{ccc|c} 6 & 2 & 2 & -2 \\ 0 & 0.0001 & -0.3333 & 1.667 \\ 0 & 0 & 5555 & -27790 \end{array}\right]$$

From this, we get that $z = \dfrac{-27790}{5555} = -5.003$

From the second equation, we get that $y = \dfrac{1.667 + 0.3333(-5.003)}{0.0001}$

$= 0$

From the first equation $x = \dfrac{-2 - 2(0) - 2(-5.003)}{6}$

$= 1.335.$

2c) We solve the system using 4 digit floating point arithmetic with rounding and partial pivoting.

$$\left[\begin{array}{ccc|c} 6 & 2 & 2 & -2 \\ 2 & 0.6667 & 0.3333 & 1 \\ 1 & 2 & -1 & 0 \end{array}\right] \underset{R_3 \leftrightarrow R_1}{\sim} \left[\begin{array}{ccc|c} 1 & 2 & -1 & 0 \\ 2 & 0.6667 & 0.3333 & 1 \\ 6 & 2 & 2 & -2 \end{array}\right]$$

$$\underset{\substack{R_2 = R_2 - 2R_1 \\ R_3 = R_3 - 6R_1}}{\sim} \left[\begin{array}{ccc|c} 1 & 2 & -1 & 0 \\ 0 & -3.333 & 2.333 & 1 \\ 0 & -10 & 8 & -2 \end{array}\right] \underset{R_3 = R_3 - \frac{10}{3.333}R_2}{\sim}$$

$$\left[\begin{array}{ccc|c} 1 & 2 & -1 & 0 \\ 0 & -3.333 & 2.333 & 1 \\ 0 & 0 & 1.001 & -5 \end{array}\right]$$

From this, we get that $z = \dfrac{-5}{1.001} = -4.995$

From the second equation, we get that $y = \dfrac{1 - 2.333(-4.995)}{-3.333}$

$= -3.795$

From the first equation, we get that $x = -2(-3.795) - 4.995$

$= 2.595.$

APPM 5600 - Homework 5

Eappen Nelluvelil

October 1, 2021

1. Let $\mathbf{A} \in \mathbb{R}^{n \times n}$ be a tridiagonal matrix where the diagonal entries are given by a_j for $j = 1, \ldots, n$, the lower diagonal entrires are b_j for $j = 2, \ldots, n$, and the upper diagonal entries are c_j for $j = 1, \ldots, n-1$.

   (a) For $n = 3$, derive the LU factorization of the matrix $\mathbf{A}$.

   (b) What is the extension of the LU factorization for general n?

   (c) Theorem 8.2 in Atkinson states that Gaussian elimination applied to a tridiagonal matrix satisfying certain diagonal-dominance conditions does not require pivoting. What is the operation count (give an exact formula) when applying Gaussian elimination to a tridiagonal system without pivoting? **You must explain how you derive the operation count.**

   When applying Gaussian elimination to a tridiagonal system that does not require pivoting, we have that in the k^{th} row, where $2 \leq k \leq n$, we need to perform the operation $R_k - \frac{\widetilde{b}_{k+1}}{\widetilde{a}_{k-1}} R_{k-1}$. This involves one subtraction, one division, and one multiplication in the k^{th} row (the entry below $\widetilde{a}_{k-1}$ can be set to 0 immediately, which avoids us having to perform the same three operations). Since we have to perform Gaussian elimination to $n-1$ rows and there are 3 operations to do, to perform Gaussian elimination on a tridiagonal system without pivoting, we have to perform $3(n-1)$ FLOPs.

2. Consider the linear system

$$\begin{aligned} 6x + 2y + 2z &= -2 \\ 2x + \frac{2}{3}y + \frac{1}{3}z &= 1 \\ x + 2y - z &= 0. \end{aligned}$$

   (a) Verify that $\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 2.6 \\ 3.8 \\ -5 \end{bmatrix}$ is the exact solution.

   We have that

$$\begin{aligned} 6(2.6) + 2(-3.8) + 2(-5) = 15.6 - 7.6 - 10 &= -2, \\ 2(2.6) + \frac{2}{3}(-3.8) + \frac{1}{3}(-5) = 5.2 + \frac{2}{3}\frac{19}{5} - \frac{1}{3}\frac{15}{3} &= 1, \\ 2.6 + 2(-3.8) + 5 &= 0, \end{aligned}$$

   i.e., the exact solution is $\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 2.6 \\ 3.8 \\ -5 \end{bmatrix}$, as desired.

   (b) Using 4-digit floating point arithmetic with rounding, solve the system via Gaussian elimination without pivoting.

   Using 4-digit floating point arithmetic with rounding, we solve the system via Gaussian elimination without pivoting to obtain that $x = 1.335$, $y = 0$, and $z = -5.003$.

(c) Repeat part (b) with partial pivoting.

Repeating part (b) with partial pivoting, we obtain that $x = 2.595$, $y = -3.795$, and $z = -4.995$.

(d) Which method is more accurate, i.e., stable?

Gaussian elimination with partial pivoting is more stable than without partial pivoting. This is because with partial pivoting, we can avoid propagating round-off errors that arise from multiplying rows by the reciprocal of small (in magnitude) pivot values.

3. Consider the system $\mathbf{Ax} = \mathbf{b}$, where

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

$$\mathbf{b} = \begin{bmatrix} 2 \\ 1 \\ 2 \\ 2 \\ 1 \\ 2 \end{bmatrix}.$$

Use the ones vector as $\mathbf{x_0}$, i.e., $\mathbf{x_0} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}^T$.

(a) Use the Gauss-Jacobi iteration to approximate the solution to this problem with $\epsilon = 10^{-7}$.

The Gauss-Jacobi iteration is given by

$$\mathbf{x_{k+1}} = \mathbf{D}^{-1} \left(\mathbf{b} - (\mathbf{L} + \mathbf{U}) \mathbf{x_k} \right),$$

where $\mathbf{D}$ is the diagonal of $\mathbf{A}$, i.e, `D = diag(diag(A))` and $(\mathbf{L} + \mathbf{U}) = \mathbf{A} - \mathbf{D}$, i.e., $\mathbf{L}$ is the strictly lower-triangular part of $\mathbf{A}$ and $\mathbf{U}$ is the strictly upper-triangular part of $\mathbf{A}$. The iteration took 40 iterations for the absolute error (in the 2-norm) of successive iterates to fall below $\epsilon = 10^{-7}$.

(b) Use the Gauss-Seidel iteration to approximate the solution to the problem with $\epsilon = 10^{-7}$.

The Gauss-Seidel iteration is given by

$$\mathbf{x_{k+1}} = (\mathbf{D} + \mathbf{L})^{-1} \left(\mathbf{b} - \mathbf{U} \mathbf{x_k} \right),$$

where $\mathbf{D}$, $\mathbf{L}$, and $\mathbf{U}$ are defined as for the Gauss-Jacobi iteration. The Gauss-Seidel iteration took 22 iterations for the absolute error (in the 2-norm) of successive iterates to fall below $\epsilon = 10^{-7}$.

(c) Use the SOR iteration with $\omega = 1.6735$ to approximate the solution to this problem with $\epsilon = 10^{-7}$.

The SOR iteration is given by

$$\mathbf{x_{k+1}} = (\mathbf{D} + \omega \mathbf{L})^{-1} \left(\omega \mathbf{b} - (\omega \mathbf{U} + (\omega - 1) \mathbf{D}) \mathbf{x_k} \right),$$

where $\mathbf{D}$, $\mathbf{L}$, and $\mathbf{U}$ are defined as for the Gauss-Jacobi iteratoin. The SOR iteration took 49 iterations for the absolute error (in the 2-norm) of successive iterates to fall below $\epsilon = 10^{-7}$ with $\omega = 1.6735$.

(d) Which method converges faster? Do you expect this to be always true?

The Gauss-Seidel iteration converged the fastest out of the three iteration schemes. However, we should not expect this to be always true, as we can pick a more optimal value for ω for the SOR iteration that will make it converge (in absolute error) in fewer iterations to a solution than either the Gauss-Jacobi or Gauss-Seidel iterations.

(e) Set $c = \rho(\mathbf{B})$ (spectral radius). Use the following error estimate to derive error bounds for the last computed approximations with all methods:

$$||\mathbf{x_{k+1}} - \mathbf{x}|| \leq \frac{c}{1-c} ||\mathbf{x_{k+1}} - \mathbf{x_k}|| \,.$$

The error bounds for the three iteration methods are given below:

i. Gauss-Jordan iteration
The spectral radius for the Gauss-Jordan iteration is approximately 0.68301, and the error bound is given by

$$\begin{aligned} ||\mathbf{x_{k+1}} - \mathbf{x}|| &\leq \frac{c}{1-c} ||\mathbf{x_{k+1}} - \mathbf{x_k}|| \\ &\approx 0.0000001669161786 \end{aligned}$$

ii. Gauss-Seidel iteration
The spectral radius for the Gauss-Seidel iteration matrix is approximately 0.48058, and the error bound is given by

$$\begin{aligned} ||\mathbf{x_{k+1}} - \mathbf{x}|| &\leq \frac{c}{1-c} ||\mathbf{x_{k+1}} - \mathbf{x_k}|| \\ &\approx 0.0000000847837315 \end{aligned}$$

iii. SOR iteration
The spectral radius for the SOR iteration matrix is approximately 0.72573, and the error bound is given by

$$\begin{aligned} ||\mathbf{x_{k+1}} - \mathbf{x}|| &\leq \frac{c}{1-c} ||\mathbf{x_{k+1}} - \mathbf{x_k}|| \\ &\approx 0.0000002139901023. \end{aligned}$$

(f) What happens if you change the parameter ω for the SOR iteration?
If we change the parameter ω for the SOR iteration, the SOR iteration converges (in absolute error) to a solution in fewer iterations than with the previous choice of $\omega = 1.6735$. For example, if we pick $\omega = 1.2$, the SOR iteration converges in 14 iterations.

4. The linear system of equations

$$\begin{bmatrix} 1 & -a \\ -a & 1 \end{bmatrix} \mathbf{x} = \mathbf{b},$$

where a is a real number, can, under certain conditions, be solved by the iterative method

$$\begin{bmatrix} 1 & 0 \\ -\omega a & 1 \end{bmatrix} \mathbf{x_{k+1}} = \begin{bmatrix} 1-\omega & \omega a \\ 0 & 1-\omega \end{bmatrix} \mathbf{x_k} + \omega \mathbf{b}.$$

(a) For which values of a is the method convergent for $\omega = 1$?
When $\omega = 1$, the iterative method is given by

$$\begin{bmatrix} 1 & 0 \\ -a & 1 \end{bmatrix} \mathbf{x_{k+1}} = \begin{bmatrix} 0 & a \\ 0 & 0 \end{bmatrix} \mathbf{x_k} + \omega \mathbf{b}.$$

The inverse of $\begin{bmatrix} 1 & 0 \\ -a & 1 \end{bmatrix}$ is given by $\begin{bmatrix} 1 & 0 \\ a & 1 \end{bmatrix}$. Multiplying by the inverse on both sides of the iteration, we get that

$$\mathbf{x_{k+1}} = \begin{bmatrix} 0 & a \\ 0 & a^2 \end{bmatrix} \mathbf{x_k} + \omega \begin{bmatrix} 1 & 0 \\ a & 1 \end{bmatrix} \mathbf{b}.$$

For this iteration to converge, the spectral radius of $\begin{bmatrix} 0 & a \\ 0 & a^2 \end{bmatrix}$ must be less than 1. The characteristic polynomial of this matrix is given by $\lambda^2 - \lambda a^2$, which is zero when $\lambda = 0$ or when $\lambda = a^2$. Thus, for the spectral radius to be less than one, it must the case that $|a| < 1$, i.e., $-1 < a < 1$.

(b) For $a = 0.5$, find the value of $\omega \in \{0.8, 0.9, 1.0, 1.1, 1.2, 1.3\}$ which minimizes the spectral radius of the matrix

$$\begin{bmatrix} 1 & 0 \\ -\omega a & 1 \end{bmatrix}^{-1} \begin{bmatrix} 1-\omega & \omega a \\ 0 & 1-\omega \end{bmatrix}.$$

Taking $a = \frac{1}{2}$, we have that $\begin{bmatrix} 1 & 0 \\ -\omega a & 1 \end{bmatrix}^{-1} = \begin{bmatrix} 1 & 0 \\ \frac{1}{2}\omega & 1 \end{bmatrix}$, and

$$\begin{bmatrix} 1 & 0 \\ -\omega a & 1 \end{bmatrix}^{-1} \begin{bmatrix} 1-\omega & \omega a \\ 0 & 1-\omega \end{bmatrix} = \begin{bmatrix} (1-\omega) & \frac{1}{2}\omega \\ \frac{1}{2}(1-\omega) & \frac{1}{4}\omega^2 + (1-\omega) \end{bmatrix}.$$

Computing the spectral radius of the above matrix for the specified values of ω, we see that the ω that minimizes the spectral radius of the matrix is $\omega = 1.1$, with corresponding spectral radius of approximately 0.1.

Problem 3

```
clc;
clear;
close all;

fprintf("Problem 3\n\n");

epsilon = 1e-7;

A = [4 -1 0 -1 0 0; ...
    -1 4 -1 0 -1 0; ...
    0 -1 4 -1 0 -1; ...
    -1 0 -1 4 -1 0; ...
    0 -1 0 -1 4 -1; ...
    0 0 -1 0 -1 4];

b = [2; 1; 2; 2; 1; 2];

D = diag(diag(A));
L_plus_U = A-D;

% Perform Gauss-Jacobi iteration
GJ_iters = 0;
x0 = ones(6, 1);
abs_err = Inf;

while abs_err >= epsilon
    x1 = (D\eye(size(D)))*(b - L_plus_U*x0);
    GJ_iters = GJ_iters + 1;
    abs_err = norm(x0-x1, 2);
    x0 = x1;
end

GJ_abs_err = abs_err;

fprintf("Gauss-Jacobi iterations required to get absolute error
 between successive iterates below %1.0e: %d\n", ...
        epsilon, GJ_iters);

% Perform Gauss-Seidel iteration
L        = tril(A);
U_strict = triu(A, 1);
GS_iters = 0;
x0 = ones(6, 1);
abs_err = Inf;

while abs_err >= epsilon
    x1 = (L\eye(size(L)))*(b - U_strict*x0);
    GS_iters = GS_iters + 1;
    abs_err = norm(x0-x1, 2);
    x0 = x1;
```

```
end

GS_abs_err = abs_err;

fprintf("Gauss-Seidel iterations required to get absolute error
 between successive iterates below %1.0e: %d\n", ...
        epsilon, GS_iters);

% Perform SOR iteration
L_strict = tril(A, -1);
omega = 1.6735;
% If we pick omega = 1.2, SOR out-performs the GJ and GS iterations,
 i.e.,
% it takes 14 iterations for the absolute error of successive iterates
 (in
% the 2-norm) to fall below 10^(-7)
% omega = 1.2;
SOR_iters = 0;
x0 = ones(6, 1);
abs_err = Inf;

while abs_err >= epsilon
    x1 = ((D+omega*L_strict)\eye(size(D)))*(omega*b - (omega*U_strict
 + (omega-1)*D)*x0);
    SOR_iters = SOR_iters + 1;
    abs_err = norm(x0-x1, 2);
    x0 = x1;
end

SOR_abs_err = abs_err;

fprintf("SOR iterations required to get absolute error between
 successive iterates below %1.0e: %d\n", ...
        epsilon, SOR_iters);

% Find error estimates for the three iteration methods
spectral_radius_B_GJ = max(abs(eig(-inv(D)*L_plus_U)));
spectral_radius_B_GS = max(abs(eig(inv(L)*U_strict)));
spectral_radius_B_SOR = max(abs(eig(inv(D
+omega*L_strict)*(omega*U_strict + (omega-1)*D))));

fprintf("\n");
fprintf("Spectral radius of B matrix for Gauss-Jacobi iteration: %0.5f
\n", spectral_radius_B_GJ);
fprintf("Spectral radius of B matrix for Gauss-Seidel iteration: %0.5f
\n", spectral_radius_B_GS);
fprintf("Spectral radius of B matrix for SOR iteration: %0.5f\n",
 spectral_radius_B_SOR);
fprintf("\n");

fprintf("\n");
fprintf("Error bound for Gauss-Jacobi iteration: %0.16f\n", ...
        (spectral_radius_B_GJ/(1-spectral_radius_B_GJ))*GJ_abs_err);
fprintf("Error bound for Gauss-Seidel iteration: %0.16f\n", ...
```

```
        (spectral_radius_B_GS/(1-spectral_radius_B_GS)*GS_abs_err));
fprintf("Error bound for SOR iteration: %0.16f\n", ...
        (spectral_radius_B_SOR/(1-
spectral_radius_B_SOR))*SOR_abs_err);
fprintf("\n");
```

```
Problem 3

Gauss-Jacobi iterations required to get absolute error between
 successive iterates below 1e-07: 40
Gauss-Seidel iterations required to get absolute error between
 successive iterates below 1e-07: 22
SOR iterations required to get absolute error between successive
 iterates below 1e-07: 49

Spectral radius of B matrix for Gauss-Jacobi iteration: 0.68301
Spectral radius of B matrix for Gauss-Seidel iteration: 0.48058
Spectral radius of B matrix for SOR iteration: 0.72573


Error bound for Gauss-Jacobi iteration: 0.0000001669161786
Error bound for Gauss-Seidel iteration: 0.0000000847837315
Error bound for SOR iteration: 0.0000002139901023
```

Problem 4

```
clear;
fprintf("\nProblem 4\n");

omegas = 0.8:0.1:1.3;

% Compute spectral radii of the iteration matrix given in problem 4
 for
% various omega values
fprintf("\n");
for i=1:length(omegas)
    iter_matrix = [(1-omegas(i)), (1/2)*omegas(i); ...
                   (1/2)*omegas(i)*(1-omegas(i)),
 (1/4)*(omegas(i))^(2)+(1-omegas(i))];
    rho_iter_matrix = max(abs(eig(iter_matrix)));
    str = sprintf("Spectral radius for iteration matrix (omega =
 %0.2f): %0.16f", ...
            omegas(i), abs(rho_iter_matrix));
    disp(str);
end
fprintf("\n");
```

```
Problem 4

Spectral radius for iteration matrix (omega = 0.80):
 0.4759591794226542
```

```
Spectral radius for iteration matrix (omega = 0.90):
 0.3758969653329253
Spectral radius for iteration matrix (omega = 1.00):
 0.2500000000000000
Spectral radius for iteration matrix (omega = 1.10):
 0.1000000000000001
Spectral radius for iteration matrix (omega = 1.20):
 0.2000000000000000
Spectral radius for iteration matrix (omega = 1.30):
 0.3000000000000000
```

Published with MATLAB® R2021a