MEMS 1060: Homework #3

Due on February 21, 2021 at 6:00pm

Professor Sammak Th 6:00PM

Made using LaTeX/Inkscape. Source files available upon request.

Shane Riley

Problem 1

Given the following linear system of equations, solve using Gauss-Jordan, LU decomposition using Crout's method, and discuss whether any iterative methods will converge to the solution for the system.

- $x_1 + 2x_2 - 2x_3 = 9$
- $2x_1 + 3x_2 + x_3 = 23$
- $3x_1 + 2x_2 - 4x_3 = 11$

Solution

First we will construct an augmented matrix in order to solve the system using Gauss-Jordan:

$$A = \left[\begin{array}{ccc|c} 1 & 2 & -2 & 9 \\ 2 & 3 & 1 & 23 \\ 3 & 2 & -4 & 11 \end{array}\right]$$

First we scale the top row such that the first nonzero element 1. This is already the case. Now we subtract the first row from the others such that the other rows contain a 0 in the first column:

$$A = \left[\begin{array}{ccc|c} 1 & 2 & -2 & 9 \\ 0 & -1 & 5 & 5 \\ 0 & -4 & 2 & -16 \end{array}\right]$$

Following the same two steps for the second row:

$$A = \left[\begin{array}{ccc|c} 1 & 2 & -2 & 9 \\ 0 & 1 & -5 & -5 \\ 0 & -4 & 2 & -16 \end{array}\right]$$

$$A = \left[\begin{array}{ccc|c} 1 & 0 & 8 & 19 \\ 0 & 1 & -5 & -5 \\ 0 & 0 & -18 & -36 \end{array}\right]$$

Once more for row 3:

$$A = \left[\begin{array}{ccc|c} 1 & 0 & 8 & 19 \\ 0 & 1 & -5 & -5 \\ 0 & 0 & 1 & 2 \end{array}\right]$$

$$A = \left[\begin{array}{ccc|c} 1 & 0 & 0 & 3 \\ 0 & 1 & 0 & 5 \\ 0 & 0 & 1 & 2 \end{array}\right]$$

Following $Ax = b$ notation, we have found that $x = [3, 5, 2]$ using Gauss-Jordan. MATLAB confirms this result

Problem 1 continued on next page. . .

Table 1: Jacobi iteration solutions

epsilon	Num Iterations	x	y	z
(True)	–	3.797665369649805	1.968871595330739	5.560311284046692
0.001	7	3.797895721743048	1.968890149835314	5.560429552510743
0.0001	8	3.797616380349643	1.968867537357994	5.560286113067821

Now using Crout's method, we can solve the same system. We analyze the first column, then first row, then second column, etc. in order to put together our L and U matrices. Then y is found using L and b, then x from U and y. The solution is implemented in the source file.

$$[A]x = b$$
$$[A] = [L][U]$$
$$[L]y = b$$
$$[U]x = y$$

Following $Ax = b$ notation, we have found that $x = [3, 5, 2]$ using Crout's LU decomposition.
Finally, we evaluate whether iterative methods will work by evaluating whether $[A]$ is diagonally dominant. We do that by ensuring the sum of the off-diagonal elements does not exceed that of the diagonal ones. Since for column 1, the off-diagonal elements sum to 5 and the diagonal element is only 1, this condition is not met. This means that the system may not converge using iterative methods.

Problem 2

Using the Jacobi iterative method, solve the system of equations. Use an initial guess of 0,0,0, using $\epsilon = 0.001$ and then $\epsilon = 0.0001$.

- $8x_1 + 2x_2 + 3x_3 = 51$
- $2x_1 + 5x_2 + x_3 = 23$
- $-3x_1 - x_2 + 6x_3 = 20$

Solution

Using MATLAB, a Jacobi iterative method is implemented to evaluate the solution iteratively. Each iteration holds the previous one such that convergence can be checked using ϵ. Additionally, the true value is calculated for reference. The A and b matrices are specified in MATLAB and passed into the solver. See the source code, and the solutions table.
Using Jacobi iteration, we find the system converges towards $x = 3.7976; y = 1.9689; z = 5.5603]$. Reducing epsilon takes an additional iteration to converge, but still converges.

Problem 3

Find the condition number of the following matrix using the infinity norm. Be sure to describe all steps taken.

$$A = \begin{bmatrix} 6 & 3 & 11 \\ 3 & 2 & 7 \\ 3 & 2 & 6 \end{bmatrix}$$

Solution

To find the infinity norm of a matrix, the absolute values of all elements in each row are summed. The largest sum is taken as the infinity norm. Doing our sums:

1. $6 + 3 + 11 = 20$
2. $3 + 2 + 7 = 12$
3. $3 + 2 + 6 = 11$

Since our largest row sum is 20, the condition number is 20. MATLAB confirms this result

```
%% MEMS 1060 Homework 3
% Author: Shane Riley
% Date: 2/21/2021
format long
%% Problem 1
% Given the following linear system, solve using GJ and Crout's method
disp(" ");
disp("Problem 1");

A = [...
    1   2  -2
    2   3   1
    3   2  -4];

b = [...
    9
    23
    11];

x_true = A\b;
% disp("x = " + x);

% Get L,U
[L, U] = myCrout3x3(A);
disp(L);
disp(U);

% Find y
y = myForward(L,b);

% Find x
x = myBackward(U,y);
disp(x);
```

```
%% Problem 2
% Use Jacobi iterative method to solve using 0,0,0 and epsilon = 0.001 and
% 0.0001
disp(" ");
disp("Problem 2");

A = [...
    8   2   3
    2   5   1
    -3  -1  6];

b = [...
    51
    23
    20];

[x_e_001, i_001] = myJacobi3x3(A,b,[0;0;0],0.001,100);
[x_e_0001, i_0001] = myJacobi3x3(A,b,[0;0;0],0.0001,100);
disp(x_e_001);
disp(i_001);
disp(x_e_0001);
disp(i_0001);

x_true=A\b;

%% Problem 3
% Find the condition number of the matrix using the infinity norm
disp(" ");
disp("Problem 3");

A = [...
    6   3   11
    3   2   7
    3   2   6];

norm = norm(A,inf);
disp(norm);


%% Supporting functions

function [L, U] = myCrout3x3(A)
% MYCROUT3x3 performs LU decomp using Crout's Method for 3x3
% A - Coefficients
% L - Lower Triangular
% U - Upper Triangular
```

```matlab

[rows, columns] = size(A);
U = zeros(rows,columns);
L = zeros(rows,columns);

for i=1:rows
    % First column of L matches A
    L(i,1)=A(i,1);

    % Diagonals for U are 1
    U(i,i) = 1;
end

% First row of U
for j=2:columns
    U(1,j) = A(1,j)/L(1,1);
end


for i=2:rows
    % Fill in L in triangular iteration
    for j=2:i
        L(i,j) = A(i,j) - L(i,1:j-1) * U(1:j-1,j);  % The product will be a
            scalar
    end
    % Fill in U in triangular iteration
    for j=i + 1:rows
        U(i,j) = (A(i,j) - L(i, 1:i-1) * U(1:i-1, j))/L(i,i);
    end
end
end

function y = myForward(L,b)
% MYFORWARD finds y using lower triangular L and b
% L - Lower Triangular
% b - Constants
% y - mid-solution

n = length(b);
y = zeros(n,1);
% Get 1,1
y(1) = b(1)/L(1,1);
for i=2:n
    y(i) = (b(i) - L(i,1:i-1)*y(1:i-1,1))./L(i,i);
end
end

function x = myBackward(U,y)
% MYFORWARD finds y using upper triangular U and y
```

```
% U - Upper Triangular
% y - mid-solution
% x - Solution

n = length(y);
x = zeros(n,1);
% Get last component
x(n) = y(n)/U(n,n);
for i=n-1:-1:1  % iterate backwards
    x(i) = (y(i) - U(i,i+1:n)*x(i+1:n,1))./U(i,i);
end
end

function [x, i] = myJacobi3x3(A,b,x0,epsilon,max_i)
% MYJACOBI3x3 finds y using upper triangular U and y
% Assumes proper sizing
% A - Coefficients
% b - constants
% x0 - Initial condition
% epsilon - convergence condition
% max_i - max iterations
% x - solution
% i - number of taken iterations

% n = 3
x_old = x0;
x = zeros(length(b),1);

% First iteration
x(1) = (b(1) - A(1,2).*x(2) - A(1,3).*x(3))./A(1,1);
x(2) = (b(2) - A(2,1).*x(1) - A(2,3).*x(3))./A(2,2);
x(3) = (b(3) - A(3,1).*x(1) - A(3,2).*x(2))./A(3,3);
i = 1;

while ~isConverged(x,x_old,epsilon)

    if (i > max_i)
        disp("Jacobi not converged");
        break;
    end
    x_old = x;
    x(1) = (b(1) - A(1,2).*x(2) - A(1,3).*x(3))./A(1,1);
    x(2) = (b(2) - A(2,1).*x(1) - A(2,3).*x(3))./A(2,2);
    x(3) = (b(3) - A(3,1).*x(1) - A(3,2).*x(2))./A(3,3);
    i = i + 1;

end

end
```

```
function b = isConverged(x, x_old, e)
% ISCONVERGED checks for convergence
% x - current iteration
% x_old - old iteration
% e - threshold

n = length(x);
b = true;

for i=1:n
    if (abs((x(i) - x_old(i))/x_old(i)) > e)
        b = false;
    end
end
end
```