Table of Contents

```
%Roshan Jaiswal-Ferri
%Section - 03
%Aero 300 Lab 4 - Iterative Methods to Solve Matrix Equations: 4/26/24
```

Workspace Prep

```
format long     %Allows for more accurate decimals
close all;      %Clears all
clear all;      %Clears Workspace
clc;            %Clears Command Window
```

PART 1:

```
tol = 1e-6;

A = [3 -1 2 ; 1 4 2 ; -1 3 5];
b = [1;1;1];
s = size(A);
u = s(1,1);
D = diag2(A);
```

PART 2: Testing for Diagonal Dominance

```
Dom = isDom(A); %Calling function below

if Dom == 1
    warning('May not Converge')
elseif Dom == 0
    disp('Matrix Will converge')
else
    disp('error (isDom)')
end
```

PART 3: Guass-Seidel

```
n=s(1,1);
m = 10000;
```

```
A2 = cat(2,A,b); %Combining to single matrix for easier calcs

%x=[0 0 0];
x = zeros(u,1); %initial guess always just a column vector of zeros

for  k = 1:10000
    err = 0;
    for i = 1 : n
        x1 = 0;
        for j = 1 : n
            x1 = x1-A2(i,j)*x(j);
        end
        x1 = (x1+A2(i,n+1))/A2(i,i);
        if abs(x1) > err
            err  = abs(x1);
        end
        x(i) = x(i) + x1;
    end

    if err <= tol
        break;
    else
    end
end

if k == 10000
    error('System did not converge after 10,000 Iterations :('); %eror was
interfering with this lol
else
    disp('Gauss-Seidel Method:');
    disp(' ')
    disp('X Vector:');
    disp(['x1: ', num2str(x(1,1))]);
    disp(['x2: ', num2str(x(2,1))]);
    disp(['x3: ', num2str(x(3,1))]);
    disp(' ');
    disp(['Solutions found in ', num2str(k), ' iterations'])
end
```

PART 4: Jacobi

```
eror = inf; %had to only use one r cuz it was interfereing with the error
function lol

x = zeros(u,1); %initial guess always just a column vector of zeros

for k = 1:10000
    x1 = D\(b - A*x); %\ built in function to solve linear equations
    x = x + x1;

    eror = max(abs(x1./x)); %Checking error by taking maximum from col vector
    if eror < tol
        break
    end
end

if k == 10000
    error('System did not converge after 10,000 Iterations :('); %eror was
interfering with this lol
else
    disp('Jacobi Method:');
    disp(' ')
    disp('X Vector:');
    disp(['x1: ', num2str(x(1,1))]);
    disp(['x2: ', num2str(x(2,1))]);
    disp(['x3: ', num2str(x(3,1))]);
    disp(' ');
    disp(['Solutions found in ', num2str(k), ' iterations'])
    disp(' ')
end
```

FUNCTION: Diagonalizing

```
function [D] = diag2(A) %Used this cuz i needed them to be in square matrix
and diag returns a column vector
    s = size(A);
    D = zeros(s);

    for i = 1:s(1,1) %Will only work with square matricis
        D(i,i) = A(i,i);
    end
end
```

FUNCTION: Checking for Dominance

```
%NOTE: Even though my randM function only generates square matracies this
will work with any size NxM
function [isDom] = isDom(A)
    s = size(A);
    isDom = 0;

    if s(1,1) < s(1,2)
        s2 = s(1,1);
    elseif s(1,1) >= s(1,2)
        s2 = s(1,2);
    end

    for i = 1:s2 %For loop testing for diagonal dominance
        x=0;
        if isDom == 1
            break
        end
        for j = 1:s(:,1)
            if j > s(1,2) || i > s(1,1) %stopping the loop if the matrix is
mxn and not nxn and it is looping past an existing column or row
                break
            end
            x = x + A(i,j);
        end
        x = x - A(i,i);
        if A(i,i) < x %if any of the rows are not dominant set isDom to not
dom
            isDom = 1;
        end
    end
end
```

FUNCTION: Rand Matrix

This is how i tested my code, it looks like it works with many matricies!

```
function [M] = randM()
    y = randi([2 9]);
    %z = randi([2 9]);
    M = randi([-10 10],y,y);
end

%Creates random matrix of size 3-5x3-5 with values ranging from -10 to 10
```

PSEUDOCODE

```
%Guass-Siedel:
% Start with initial guess of all zero loop through solving a system of
% equations with the guesses one at a time. Continue until iteration limit
% reached or it is within tolerance

%JACOBI:
% Start with an initial guess for each iteration it updates based on the
prevuious
% answer based off the diag of A and b matrix, by calculating the system of
% equations.
```

Published with MATLAB® R2023b