# Table of Contents



```
% AERO 300 - Lab 4 - Joshua Oates
% sections 1 and 2 will use test code provided in class examples, section 3
% will use my own test cases
```

# Section 0 - clean up

```
clear all
close all
clc
```

# Section 1 - test Jacobi against same test cases as used in example

```
disp(" ")
disp("-------------Section 1-------------")
disp(" ")


%For all cases, set the tolearance to
TOL = 0.5 *10^(-6);


% Case 1: A is strictly diagonally row dominant
A = [4, 2;...
     -1, 2];

b = [3; 2];
x_0 = [1; 1];
[x1, X1, k1] = JoshJacobi(A, b, x_0, TOL);
% Case 2: A is NOT strictly diagonally dominant but Jacobi converges


A = [10, -2, -1;...
      -1,  5,  3;...
       2,  2, -2];

b = [1; 2; 3];
x_0 = [1; 1; 1];
[x2, X2, k2] = JoshJacobi(A, b, x_0, TOL);
```

```
% Case 3: A is NOT strictly diagonally dominant and Jacobi does not converge
A = [10, -2, -1;...
      -1, -5,  3;...
       2,  2, -2];

b = [1; 2; 3];
x_0 = [1; 1; 1];
[x3, X3, k3] = JoshJacobi(A, b, x_0, TOL);
[x4, X4, k4] = JoshJacobi(A, b, x_0, TOL,200);

str = "In " + k1 + " iterations x1 was found to be: ";
disp(str)
disp(num2str(x1,'%.7f'))

str = "In " + k2 + " iterations x2 was found to be: ";
disp(str)
disp(num2str(x2,'%.7f'))

str = "In " + k3 + " (defualt) iterations x3 was found to not converge,
 However;";
disp(str)

str = "In " + k4 + " iterations x3 was found to be: ";
disp(str)
disp(num2str(x4,'%.7f'))


-------------Section 1-------------

Warning: A is not strictly diagonally row dominant, it may not converge.
Warning: A is not strictly diagonally row dominant, it may not converge.
Warning: Convergence failed
Warning: A is not strictly diagonally row dominant, it may not converge.
In 23 iterations x1 was found to be:
0.1999998
1.1000000
In 46 iterations x2 was found to be:
 0.1987180
 0.7628203
-0.5384613
In 100 (defualt) iterations x3 was found to not converge, However;
In 117 iterations x3 was found to be:
-1.7083325
-4.9583313
-8.1666634
```

# Section 2 - test GuassSeidel against same test cases as used in example

```
disp(" ")
disp("-------------Section 2-------------")
```

```
disp(" ")

%For all cases, set the tolearance to
TOL = 0.5 *10^(-6);


% Case 1: A is strictly diagonally row dominant
A = [4, 2;...
     -1, 2];

b = [3; 2];
x_0 = [1; 1];
[x1, X1, k1] = JoshGuassSeidel(A, b, x_0, TOL);
% Case 2: A is NOT strictly diagonally dominant but Jacobi converges


A = [10, -2, -1;...
      -1,  5,  3;...
       2,  2, -2];

b = [1; 2; 3];
x_0 = [1; 1; 1];
[x2, X2, k2] = JoshGuassSeidel(A, b, x_0, TOL);


% Case 3: A is NOT strictly diagonally dominant and Jacobi does not converge
A = [10, -2, -1;...
     -1, -5,  3;...
      2,  2, -2];

b = [1; 2; 3];
x_0 = [1; 1; 1];
[x3, X3, k3] = JoshGuassSeidel(A, b, x_0, TOL);

str = "In " + k1 + " iterations x1 was found to be: ";
disp(str)
disp(num2str(x1,'%.7f'))

str = "In " + k2 + " iterations x2 was found to be: ";
disp(str)
disp(num2str(x2,'%.7f'))

str = "In " + k3 + " iterations x3 was found to be: ";
disp(str)
disp(num2str(x3,'%.7f'))


------------Section 2-------------

Warning: A is not strictly diagonally row dominant, it may not converge.
Warning: A is not strictly diagonally row dominant, it may not converge.
In 12 iterations x1 was found to be:
0.2000000
1.1000000
```

```
In 17 iterations x2 was found to be:
 0.1987179
 0.7628206
-0.5384615
In 67 iterations x3 was found to be:
-1.7083328
-4.9583321
-8.1666648
```

# Section 3 - test of both against orriginal test cases

```
disp(" ")
disp("-------------Section 3-------------")
disp(" ")

%For all cases, set the tolearance to
TOL = 0.5 *10^(-6);


% Case 1
A = [0, 0;...
     0, 0];

b = [3; 2];
x_0 = [1; 1];
[x1J, ~, k1J] = JoshJacobi(A, b, x_0, TOL);
[x1G, ~, k1G] = JoshGuassSeidel(A, b, x_0, TOL);


% Case 2
A = [1, 0, 0;...
     0, 1, 0;...
     0, 0, 1];

b = [1; 2; 3];
x_0 = [0; 0; 0];
[x2J, ~, k2J] = JoshJacobi(A, b, x_0, TOL);
[x2G, ~, k2G] = JoshGuassSeidel(A, b, x_0, TOL);


% Case 3
A = [ 10, -2, -1;...
     -1,  3,  3;...
      2,  2, -2];

b = [1; 2; 3];
x_0 = [1; 1; 1];
[x3J, ~, k3J] = JoshJacobi(A, b, x_0, TOL,10000);
[x3G, ~, k3G] = JoshGuassSeidel(A, b, x_0, TOL);

disp(" ")
```

```
disp("Case 1")
disp("Jacobi:")
disp("  iterations:")
disp(num2str(k1J))
disp("  x:")
disp(num2str(x1J))
disp(" ")
disp("GuassSeidel:")
disp("  iterations:")
disp(num2str(k1G))
disp("  x:")
disp(num2str(x1G))

disp(" ")
disp("as expected, both failed")

disp(" ")
disp("Case 2")
disp("Jacobi:")
disp("  iterations:")
disp( num2str(k2J))
disp("  x:")
disp( num2str(x2J))
disp(" ")
disp("GuassSeidel:")
disp("  iterations:")
disp(num2str(k2G))
disp("  x:")
disp(num2str(x2G))

disp(" ")
disp("as expected, both quickly succeeded")

disp(" ")
disp("Case 3")
disp("Jacobi:")
disp("  iterations:")
disp( num2str(k3J))
disp("  x:")
disp( num2str(x3J))
disp(" ")
disp("GuassSeidel:")
disp("  iterations:")
disp(num2str(k3G))
disp("  x:")
disp(num2str(x3G))

disp(" ")
disp("although it wasn't garaunteed, both converged. GuassSeidel did so much
 quicker")


----------Section 3----------
```

```
Warning: A is not strictly diagonally row dominant, it may not converge.
Warning: A has zero(s) in its diagonal,  it may not converge.
Warning: A is not strictly diagonally row dominant, it may not converge.
Warning: A has zero(s) in its diagonal,  it may not converge.
Warning: A is not strictly diagonally row dominant, it may not converge.
Warning: A is not strictly diagonally row dominant, it may not converge.

Case 1
Jacobi:
  iterations:
3
  x:
NaN
NaN

GuassSeidel:
  iterations:
2
  x:
Inf
NaN

as expected, both failed

Case 2
Jacobi:
  iterations:
3
  x:
1
2
3

GuassSeidel:
  iterations:
3
  x:
1
2
3

as expected, both quickly succeeded

Case 3
Jacobi:
  iterations:
219
  x:
   0.275
 0.99167
-0.23333

GuassSeidel:
  iterations:
```

```
22
  x:
    0.275
 0.99167
-0.23333

although it wasn't garaunteed, both converged. GuassSeidel did so much quicker
```

*Published with MATLAB® R2022a*

```
function [x,X,k] = JoshJacobi(A,b,x0,TOL,maxI)
% take A matrix
% take b vector
% take x0
arguments
    A{mustBeNumeric,mustBeReal}
    b{mustBeNumeric,mustBeReal,mustBeVector}
    x0{mustBeNumeric,mustBeReal,mustBeVector}
    TOL(1,1) {mustBeNumeric,mustBeReal,mustBePositive} = .01
    maxI (1,1) {mustBeNumeric,mustBeReal,mustBePositive,mustBeInteger} = 100
end
% create defualt maxI
% measure n as dimension
[n,m] = size(A);
% verify they are all same dim and A is square
if n ~= m
    error("A must be square matrix.")
elseif length(b) ~= n
    error("b must be the same size as A.")
elseif length(x0) ~= n
    error("x0 must be the same size as A.")
end

clear m % m and n have been verified to contain the same value
% test if it is SDRD as a test for convergece
myWarnings = "";
for i = 1:n % step through each row
    r = A(i,:); % get row in question
    d = abs(r(i)); % get the magnitude of the diagonal (row i, col i)
    s = sum(abs(r)) - d; % get the sum of the rest of the row
    if d <= s
        myWarnings(1) = "A is not strictly diagonally row dominant, it may not
 converge.";
    end
    if d == 0
        myWarnings(2) = "A has zero(s) in its diagonal,  it may not
 converge.";
    end
end
% send warning if it wont neccisarily converge
for warn = myWarnings
    if warn ~= ""
        warning(warn)
    end
end
clear d s r myWarning warn i % clear vars from SDRD check

% do jacobi iteration, use k as iterator and i , j as indicies
% X will be 2 dimensional with indexs (i,k)
% A will be 2 dimensional with indexs (i,j)
% b will be 1 dimensional with index (i)
% the value of X(i,k+1) is given by
```

```
% X(i,k+1) =( 1/A(i,i) ) * ( b(i) - ( (sumOverj ( A(i,j) * X(j,k) )- A(i,i) *
 X(i,k))
k = 1;
X(:,k) = x0;
err = inf;
while (err>TOL) & (k < maxI) % run until TOL or maxI met
    for i = 1:n % for each row
        s = 0;
        for j = 1:n % for each j
             s = s + A(i,j) * X(j,k); % sumOverj ( A(i,j) * X(j,k)
        end
        X(i,k+1) =( 1/A(i,i) ) * ( b(i) - (s - A(i,i) * X(i,k) ) );
    end
    k = k + 1;
    err = norm( (X(:,k)-X(:,k-1)) ,inf);
end
x = X(:,k);

if k >= maxI
    x = [];
    warning("Convergence failed")
end
```

*Published with MATLAB® R2022a*

```
function [x,X,k] = JoshGuassSeidel(A,b,x0,TOL,maxI)
% take A matrix
% take b vector
% take x0
arguments
    A{mustBeNumeric,mustBeReal}
    b{mustBeNumeric,mustBeReal,mustBeVector}
    x0{mustBeNumeric,mustBeReal,mustBeVector}
    TOL(1,1) {mustBeNumeric,mustBeReal,mustBePositive} = .01
    maxI (1,1) {mustBeNumeric,mustBeReal,mustBePositive,mustBeInteger} = 100
end
% create defualt maxI
% measure n as dimension
[n,m] = size(A);
% verify they are all same dim and A is square
if n ~= m
    error("A must be square matrix.")
elseif length(b) ~= n
    error("b must be the same size as A.")
elseif length(x0) ~= n
    error("x0 must be the same size as A.")
end

clear m % m and n have been verified to contain the same value
% test if it is SDRD as a test for convergece
myWarnings = "";
for i = 1:n % step through each row
    r = A(i,:); % get row in question
    d = abs(r(i)); % get the magnitude of the diagonal (row i, col i)
    s = sum(abs(r)) - d; % get the sum of the rest of the row
    if d <= s
        myWarnings(1) = "A is not strictly diagonally row dominant, it may not
 converge.";
    end
    if d == 0
        myWarnings(2) = "A has zero(s) in its diagonal,  it may not
 converge.";
    end
end
% send warning if it wont neccisarily converge
for warn = myWarnings
    if warn ~= ""
        warning(warn)
    end
end
clear d s r myWarning warn i % clear vars from SDRD check

% do GuassSeidel iteration, use k as iterator and i , j as indicies
% X will be 2 dimensional with indexs (i,k)
% A will be 2 dimensional with indexs (i,j)
% b will be 1 dimensional with index (i)
% the value of X(i,k+1) is given by
```

```
% X(i,k+1) =( 1/A(i,i) ) * ( b(i) - ( (sumOverjUpToi ( A(i,j) * X(j,k+1) )-
 sumOverjAfteri ( A(i,j) * X(j,k))
k = 1;
X(:,k) = x0;
err = inf;
while (err>TOL) & (k < maxI) % run until TOL or maxI met
    for i = 1:n % for each row
        s1 = 0;
        for j = 1:i-1 % for each j up to the one before i (exist in the k+1
 depth plane)
             s1 = s1 + A(i,j) * X(j,k+1); % sumOverj ( A(i,j) * X(j,k+1)
        end
        s2 = 0;
        for j= i+1:n % for each j up to the one after i (exist in the k depth
 plane but not k+1)
           s2 = s2 + A(i,j) * X(j,k); % sumOverj ( A(i,j) * X(j,k)
        end
        X(i,k+1) =( 1/A(i,i) ) * ( b(i) - s1 - s2 );
    end
    k = k + 1;
    err = norm( (X(:,k)-X(:,k-1)) ,inf);
end
x = X(:,k);

if k >= maxI
    x = [];
    warning("Convergence failed")
end
```

*Published with MATLAB® R2022a*