Table of Contents

Problem 1 1
Problem 2 2
Problem 3 3
Problem 4 5
Problem 5 5
Jacobi's Method 6
Gauss-Seidel Method 8
Check Tolerance Function 9

Problem 1

For the linear system:

```
clear
clc
A = [ 10, -1,  2,  0;
      -1, 11, -1,  3;
       2, -1, 10, -1;
       0,  3, -1.  8];

b = [6, 25, -11, 15]';
x0 = zeros(4,1);
tolerance = 10^-3;

%(a) use the Jacobi method to solve the linear system
[solJacobi, nJacobi] = JacobisMethod(A, b, x0, tolerance)

%(b) repeat part (a) using the Gauss-Seidel method
[solGS, nGS] = GSMethod(A, b, x0, tolerance)

solExact = A\b
```

```
solGS =

    1.0001
    2.0000
   -1.0000
    1.0000


nGS =

     6


solExact =
    1.0000
    2.0000
   -1.0000
    1.0000
```

Problem 2

For the linear system

```
A = [ 4,  -1,  0,  0,   0,  0;
     -1,  4, -1,  0,   0,  0;
      0, -1,  4,  0,   0,  0;
      0,  0,  0,  4,  -1,  0;
      0,  0,  0, -1,   4, -1;
      0,  0,  0,  0,  -1,  4];

b = [0, 5, 0, 6, -2, 6]';
x0 = zeros(6,1);
tolerance = 10^-4;

%(a) use the Jacobi method to solve the linear system
[solJacobi, nJacobi] = JacobisMethod(A, b, x0, tolerance)

%(b) repeat part (a) using the Gauss-Seidel method
[solGS, nGS] = GSMethod(A, b, x0, tolerance)

solExact = A\b
```

```
solGS =

    0.3571
    1.4286
    0.3571
    1.5714
    0.2857
    1.5714


nGS =

     8


solExact =

    0.3571
    1.4286
    0.3571
    1.5714
    0.2857
    1.5714
```

Problem 3

```
n = 80;
A = zeros(n);

for i = 1: n
    for j = 1: n
        if i == j
            A(i,j) = i;
        elseif ( j == i + 2 ) && ( 1 <= i ) && ( i <= n-2 )
            A(i,j) = 0.5 * i;
        elseif ( j == i - 2 ) && ( 3 <= i ) && ( i <= n   )
            A(i,j) = 0.5 * i;
        elseif ( j == i + 4 ) && ( 1 <= i ) && ( i <= n-4 )
            A(i,j) == 0.25*i;
        elseif ( j == i - 4 ) && ( 5 <= i ) && ( i <= n   )
            A(i,j) == 0.25*i;
        else
            A(i,j) = 0;
        end
    end
end

x0 = zeros(n,1);

b = x0 + pi;

tolerance = 1e-5;

[sol, n, Tj] = JacobisMethod(A, b, x0, tolerance);
sol

[sol, n, Tgs] = GSMethod(A, b, x0, tolerance);
sol
```

```
sol =

    4.8352
    2.1386
   -3.3872
   -1.1355
    4.0336
    1.7033
   -3.4234
   -1.2238
    3.7108
    1.5298
   -3.3000
   -1.2075
    3.4606
    1.4088
   -3.1378
```

```
-1.1613
 3.2339
 1.3064
-2.9605
-1.1026
 3.0178
 1.2129
-2.7759
-1.0376
 2.8073
 1.1242
-2.5873
-0.9691
 2.6002
 1.0384
-2.3964
-0.8983
 2.3953
 0.9546
-2.2039
-0.8261
 2.1921
 0.8721
-2.0105
-0.7528
 1.9900
 0.7906
-1.8164
-0.6789
 1.7889
 0.7099
-1.6218
-0.6044
 1.5885
 0.6298
-1.4270
-0.5295
 1.3887
 0.5501
-1.2319
-0.4544
 1.1894
 0.4708
-1.0367
-0.3790
 0.9905
 0.3918
-0.8414
-0.3034
 0.7920
 0.3131
-0.6459
-0.2277
 0.5937
```

```
    0.2346
   -0.4504
   -0.1518
    0.3956
    0.1563
   -0.2548
   -0.0758
    0.1978
    0.0781
   -0.0591
    0.0002
```

Problem 4

```
clc
A = [ 2, -1, 1;
      2,  2, 2;
     -1, -1, 2];

b = [1, 2, -1]';

x0 = zeros(3,1);

tolerance = 1e-5;

[sol, n, Tj] = JacobisMethod(A, b, x0, tolerance);
egnVlsTj = eig(Tj);
rhoOfTj = max(abs(egnVlsTj))

[sol, n, Tgs] = GSMethod(A, b, x0, tolerance);
egnVlsTgs = eig(Tgs);
rhoOfTj = max(abs(egnVlsTgs))
```

```
rhoOfTj =

    0.5000
```

Problem 5

```
clc
A = [ 1, 2, -2;
      1, 1,  1;
      2, 2,  1];

b = [7, 2, 5]';

x0 = zeros(3,1);

tolerance = 1e-5;
```

```
[sol, n, Tj] = JacobisMethod(A, b, x0, tolerance);
egnVlsTj = eig(Tj);
rhoOfTj = max(abs(egnVlsTj))

[sol, n, Tgs] = GSMethod(A, b, x0, tolerance);
egnVlsTgs = eig(Tgs);
rhoOfTgs = max(abs(egnVlsTgs))
```

Jacobi's Method

```
function [sol, n, Tj] = JacobisMethod(A, b, x0, tolerance)
N = diag(diag(A));
P = N - A;

xn = x0;
error = 999999;
n = 1; % number of iterations
Tj = inv(N)*P;
c = inv(N)*b;
while error > tolerance; % TODO update to for loop with maxIter
xnPlus1 = Tj*xn + c;
error = CheckTolerance(xn, xnPlus1);
n = n+1;
xn = xnPlus1;
sol = xnPlus1;
end
end
```

```
solJacobi =

    0.3571
    1.4286
    0.3571
    1.5714
    0.2857
    1.5714


nJacobi =

    12



sol =

    4.8358
    2.1388
   -3.3884
   -1.1360
    4.0354
    1.7040
   -3.4257
```

```
-1.2248
 3.7137
 1.5309
-3.3035
-1.2088
 3.4645
 1.4103
-3.1422
-1.1630
 3.2388
 1.3083
-2.9658
-1.1047
 3.0235
 1.2151
-2.7820
-1.0400
 2.8136
 1.1266
-2.5940
-0.9717
 2.6070
 1.0411
-2.4035
-0.9011
 2.4025
 0.9574
-2.2113
-0.8289
 2.1995
 0.8750
-2.0179
-0.7557
 1.9974
 0.7935
-1.8237
-0.6817
 1.7961
 0.7127
-1.6289
-0.6071
 1.5954
 0.6324
-1.4336
-0.5321
 1.3951
 0.5525
-1.2380
-0.4567
 1.1951
 0.4730
-1.0421
-0.3811
 0.9955
```

```
    0.3937
   -0.8459
   -0.3052
    0.7960
    0.3147
   -0.6496
   -0.2291
    0.5968
    0.2358
   -0.4530
   -0.1528
    0.3977
    0.1571
   -0.2564
   -0.0765
    0.1988
    0.0785
   -0.0596
    0.0000


rhoOfTj =

    1.1180


rhoOfTj =

   1.2332e-05
```

Gauss-Seidel Method

```
function [sol, n, Tgs] = GSMethod(A, b, x0, tolerance)
D = diag(diag(A));
L = -(tril(A) - D);
U = -(triu(A) - D);

Tgs = inv(D-L)*U;
c = inv(D-L)*b;

xn = x0;
error = 999999;
n = 1; % number of iterations

while error > tolerance % TODO update to for loop with maxIter
xnPlus1 = Tgs*xn + c;
error = CheckTolerance(xn, xnPlus1);
n = n+1;
xn = xnPlus1;
end
```

```
sol = xnPlus1;

end
```

Check Tolerance Function

```
function error = CheckTolerance(xn, xnPlus1)
error = norm( xnPlus1 - xn, Inf)/norm(xnPlus1, Inf);
end
```

```
solJacobi =

    0.9997
    2.0004
   -1.0004
    1.0006


nJacobi =

    10
```

Published with MATLAB® R2018b