Table of Contents

Lab12Simons.m

```
Madilyn Simons
```

```
clear; clc;
```

Problem 1

```
a) Use the Jacobi mehod to solve the linear system
b) Use the Gauss-Seidel method to solve the system
```

```
% number of unknowns / equations
n = 4;

% Coefficient matrix
A = [10 -1 2 0; ...
    -1 11 -1 3; ...
    2 -1 10 -4; ...
    0 3 -1 8];

% constant terms
b = [6; 25; -11; 15];

% x(0)
x0 = zeros(n, 1);

% tolerance
TOL = 10^-3;

% find the solution using the Jacobi method
j = jacobi(A, b, x0, TOL, n);

% find solution using the Gauss-Siedel method
g = gauss_siedel(A, b, x0, TOL, n);

% print the results
fprintf('Jacobi Approximation:\n');
print_array(j, n);

fprintf('\n');
```

```
fprintf('Gauss-Siedel Approximation:\n');
print_array(g, n);

fprintf('\n');
```

```
Jacobi Approximation:
0.935871
2.014068
-0.671121
1.035486

Gauss-Siedel Approximation:
0.935708
2.014275
-0.671415
1.035720
```

Problem 2

```
% number of unknowns / equations
n = 6;

% coefficient matrix
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];

% constant terms
b = [0; 5; 0; 6; -2; 6];

% x(0)
x0 = zeros(1, n);

% tolderance
TOL = 10^-4;

% find the solution using the Jacobi method
j = jacobi(A, b, x0, TOL, n);

% find the solution using the Gauss-Siedel method
g = gauss_siedel(A, b, x0, TOL, n);

% print the results
fprintf('Jacobi Approximation:\n');
print_array(j, n);

fprintf('\n');

fprintf('Gauss-Siedel Approximation:\n');
```

```
print_array(g, n);

fprintf('\n');
```

Jacobi Approximation:
0.357132
1.428566
0.357132
1.571426
0.285690
1.571426

Gauss-Siedel Approximation:
0.357141
1.428571
0.357143
1.571425
0.285713
1.571428

Problem 3

```
% number of unknowns / equations
n = 80;

% initialize coefficient matrix
A = zeros(n);

% initialize constant terms
b = zeros(n, 1);

% x(0)
x0 = zeros(n, 1);

% tolerance
TOL = 10^-5;

% set values for coefficient matrix and constant terms
for i=1:n
    for j=1:n
      if j == i
         A(i, j) = 2 * i;
     elseif j == i + 2 & i >= 1 & (i <= 78)
         A(i, j) = 0.5 * i;
     elseif j == i - 2 & i >= 3 & i <= 80
         A(i, j) = 0.5 * i;
     elseif j == i + 4 & i >= 1 & i <= 76
         A(i, j) = 0.25 * i;
     elseif j == i - 4 & i >= 5 & i <= 80
         A(i, j) = 0.25 * i;
     else
         A(i, j) = 0;
```

```
      end
    end

    b(i) = pi;
end

% find the solution using the Jacobi method
j = jacobi(A, b, x0, TOL, n);

% find the solution using the Gauss-Siedel method
g = gauss_siedel(A, b, x0, TOL, n);

% print the results
fprintf('Jacobi Approximation:\n');
print_array(j, n);

fprintf('\n');

fprintf('Gauss-Siedel Approximation:\n');
print_array(g, n);

fprintf('\n');
```

```
Jacobi Approximation:
1.538735
0.731422
0.107971
0.173285
0.040559
0.085250
0.166450
0.121982
0.101253
0.090460
0.072032
0.070266
0.068758
0.063247
0.059715
0.055712
0.051879
0.049249
0.046782
0.044487
0.042469
0.040538
0.038773
0.037182
0.035709
0.034351
0.033095
0.031922
0.030830
0.029810
```

0.028855
0.027959
0.027118
0.026325
0.025577
0.024870
0.024201
0.023567
0.022966
0.022394
0.021850
0.021332
0.020838
0.020366
0.019915
0.019483
0.019070
0.018674
0.018294
0.017929
0.017578
0.017241
0.016917
0.016604
0.016303
0.016012
0.015732
0.015461
0.015200
0.014947
0.014702
0.014465
0.014236
0.014013
0.013803
0.013594
0.013385
0.013188
0.012972
0.012787
0.012703
0.012527
0.012377
0.012210
0.011290
0.011141
0.012173
0.012018
0.015429
0.015238

Gauss-Siedel Approximation:
1.538733
0.731420

0.107969
0.173283
0.040556
0.085248
0.166447
0.121979
0.101249
0.090457
0.072028
0.070263
0.068754
0.063243
0.059711
0.055708
0.051874
0.049245
0.046778
0.044483
0.042465
0.040534
0.038769
0.037178
0.035705
0.034347
0.033092
0.031919
0.030826
0.029807
0.028852
0.027956
0.027115
0.026322
0.025574
0.024867
0.024199
0.023565
0.022963
0.022392
0.021848
0.021330
0.020835
0.020364
0.019913
0.019481
0.019068
0.018672
0.018292
0.017927
0.017576
0.017239
0.016915
0.016602
0.016301
0.016011

```
0.015731
0.015460
0.015199
0.014946
0.014701
0.014464
0.014234
0.014012
0.013802
0.013594
0.013384
0.013188
0.012971
0.012786
0.012703
0.012527
0.012377
0.012210
0.011290
0.011141
0.012173
0.012017
0.015429
0.015238
```

Problem 4

```
% number of unknowns / equations
n = 3;

% tolerance
TOL = 10^-5;

% x(0)
x0 = zeros(n, 1);

% coefficient matrix
A = [2 -1 1; ...
    2 2 2; ...
    -1 -1 2];

% constant terms
b = [-1; 4; -5];

% diagonal matrix
D = [2 0 0; ...
    0 2 0; ...
    0 0 2];

% lower triangular matrix
L = [0 0 0; ...
    -2 0 0; ...
```

```
    1 1 0];

% upper triangular matrix
U = [0 1 -1; ...
    0 0 -2; ...
    0 0 0];

% solve for Tj and Tg
Tj = inv(D) * (L + U);
Tg = inv(D - L) * U;

% find the spectral radii of Tj and Tg
fprintf('p(Tj) = %.2f\n', max(abs(eig(Tj))));
fprintf('\n');
fprintf('p(Tg) = %.2f\n', max(abs(eig(Tg))));
fprintf('\n');

% Since p(Tg) < 1,
% use the Gauss-Siedel method to solve the system
g = gauss_siedel(A, b, x0, TOL, n);

% print the results
fprintf('Gauss-Siedel Approximation:\n');
print_array(g, n);

fprintf('\n');
```

```
p(Tj) = 1.12

p(Tg) = 0.50

Gauss-Siedel Approximation:
1.000002
1.999997
-1.000000
```

Problem 5

```
% number of unknowns / equations
n = 3;

% tolerance
TOL = 10^-5;

% x(0)
x0 = zeros(n, 1);

% coefficient matrix
A = [1 2 -2; ...
    1 1 1; ...
    2 2 1];
```

```
% constant terms
b = [7; 2; 5];

% diagonal matrix
D = [1 0 0; ...
    0 1 0; ...
    0 0 1];

% lower triangular matrix
L = [0 0 0; ...
    -1 0 0; ...
    -2 -2 0];

% upper triangular matrix
U = [0 -2 2; ...
    0 0 -1; ...
    0 0 0];

% solve for Tj and Tg
Tj = inv(D) * (L + U);
Tg = inv(D - L) * U;

% find the spectral radii of Tj and Tg
fprintf('p(Tj) = %.2f\n', max(abs(eig(Tj))));
fprintf('\n');
fprintf('p(Tg) = %.2f\n', max(abs(eig(Tg))));
fprintf('\n');

% Since p(Tj) < 1,
% use the Jacobi method to solve the system
j = jacobi(A, b, x0, TOL, n);

% print the results
fprintf('Jacobi Approximation:\n');
print_array(j, n);

fprintf('\n');
```

```
p(Tj) = 0.00

p(Tg) = 2.00

Jacobi Approximation:
1.000000
2.000000
-1.000000
```

Published with MATLAB® R2018b