Contents

Introduction

```
%Aldous George
%EP 501
%Project 2
%This code contains excerpts from codes provided by Dr. Zettergen.
%https://github.com/Zettergren-Courses/EP501_matlab/blob/master/...
%linear_algebra
clc
clearvars
close all
```

Problem 1

```
%a Kindly refer to DLUfactor function

%b
disp('b)');
load 'testproblem.mat'
[L,U] = DLUfactor(A);

%Forward sub for b'
bprime=LTriForwardSub(L,b);

%Back substitution for x
x=backsub(cat(2,U,bprime));

disp('Doolittle LU factorisation, b:  ');
disp(x);

disp('Matlab,GNU/Octave built-in solution:  ');
disp(A\b);
```

```
b)
Doolittle LU factorisation, b:
    1.0000
    2.0000
    3.0000
    4.0000
    5.0000
    6.0000
    7.0000
    8.0000

Matlab,GNU/Octave built-in solution:
    1.0000
    2.0000
    3.0000
    4.0000
    5.0000
```

```
    6.0000
    7.0000
    8.0000
```

1 C)

```
%forward sub for b2
%Forward sub for b'
b2prime=LTriForwardSub(L,b2);

%Back substitution for x
x=backsub(cat(2,U,b2prime));

disp('Doolittle LU factorisation, b2:  ');
disp(x);

disp('Matlab,GNU/Octave built-in solution:  ');
disp(A\b2);

%forward sub for b3
%Forward sub for b'
b3prime=LTriForwardSub(L,b3);

%Back substitution for x
x=backsub(cat(2,U,b3prime));

disp('Doolittle LU factorisation, b3:  ');
disp(x);

disp('Matlab,GNU/Octave built-in solution:  ');
disp(A\b3);
```

```
Doolittle LU factorisation, b2:
    2.0000
    4.0000
    6.0000
    8.0000
   10.0000
   12.0000
   14.0000
   16.0000

Matlab,GNU/Octave built-in solution:
    2.0000
    4.0000
    6.0000
    8.0000
   10.0000
   12.0000
   14.0000
   16.0000

Doolittle LU factorisation, b3:
   10.0000
   20.0000
   30.0000
   40.0000
   50.0000
   60.0000
   70.0000
   80.0000

Matlab,GNU/Octave built-in solution:
   10.0000
```

```
   20.0000
   30.0000
   40.0000
   50.0000
   60.0000
   70.0000
   80.0000
```

1 D)

```
%d
%Calculating Inverse one column at a time
nref=length(b);
InvA=[];
for ir=1:nref
    B=zeros(nref,1);
    B(ir) = 1;
    %Forward sub for b'
    Bprime=LTriForwardSub(L,B);
    %Back substitution for x
    x=backsub(cat(2,U,Bprime)); %ith column of the Inverse
    InvA=cat(2,InvA,x);
end %for
disp('Inverse of A:  ');
disp(InvA);

disp('Matlab,GNU/Octave built-in solution:  ');
disp(inv(A));
```

```
Inverse of A:
  Columns 1 through 7

   -0.4480    0.3835    0.0281   -0.0881   -0.5795    1.0474   -0.5356
   -0.0540   -0.1948   -0.2456   -0.6264    0.1978   -0.2692    0.2222
    0.2062   -0.1064   -0.3766   -1.1154   -0.0220    0.5605    0.2837
   -0.3250    0.4251    0.0724   -0.1670   -0.3128    0.8816    0.4305
   -0.0697   -0.5582   -0.4000   -1.3059    0.0704    0.6537    0.8908
    0.3565    0.3345    0.1079   -0.1491    0.2014    0.0363   -0.2920
   -0.1222    0.1436    0.0008    0.7677   -0.2421   -0.0132   -0.1231
    0.1043   -0.2818   -0.2839   -0.2878    0.4281   -0.1212    0.1503

  Column 8

    0.2581
    0.2324
    0.3873
    0.2608
    0.6467
   -0.6463
   -0.7433
    0.0735

Matlab,GNU/Octave built-in solution:
  Columns 1 through 7

   -0.4480    0.3835    0.0281   -0.0881   -0.5795    1.0474   -0.5356
   -0.0540   -0.1948   -0.2456   -0.6264    0.1978   -0.2692    0.2222
    0.2062   -0.1064   -0.3766   -1.1154   -0.0220    0.5605    0.2837
   -0.3250    0.4251    0.0724   -0.1670   -0.3128    0.8816    0.4305
   -0.0697   -0.5582   -0.4000   -1.3059    0.0704    0.6537    0.8908
    0.3565    0.3345    0.1079   -0.1491    0.2014    0.0363   -0.2920
   -0.1222    0.1436    0.0008    0.7677   -0.2421   -0.0132   -0.1231
    0.1043   -0.2818   -0.2839   -0.2878    0.4281   -0.1212    0.1503
```

```
  Column 8

    0.2581
    0.2324
    0.3873
    0.2608
    0.6467
   -0.6463
   -0.7433
    0.0735
```

Problem 2

```
%a Kindly refer to SoR.m function

%b
disp('2 B)');
%Initialisation
load 'iterative_testproblem.mat'
nref=size(Ait,1);
nit=10;
x0=zeros(nref,1);
tol=1e-10;
w=1.1;
%Testing Successive over-Relaxation
[xit,iter]=SoR(x0,Ait,bit,tol,false,w);
disp('Solution with Successive over-Relaxation iteration: ')
disp(xit);
disp('Number of Iterations required: ')
disp(iter);
disp('Tolerance: ')
disp(tol);
disp('MATLAB built-in solution: ')
disp(Ait\bit);
```

```
2 B)
Solution with Successive over-Relaxation iteration:
    0.0329
    0.1316
    0.2400
    0.3375
    0.4142
    0.4642
    0.4839
    0.4720
    0.4293
    0.3584
    0.2641
    0.1526
    0.0310
   -0.0926
   -0.2101
   -0.3138
   -0.3971
   -0.4544
   -0.4819
   -0.4780
   -0.4427
   -0.3785
   -0.2896
   -0.1817
   -0.0619
    0.0619
    0.1817
```

```
    0.2896
    0.3785
    0.4427
    0.4780
    0.4819
    0.4544
    0.3971
    0.3138
    0.2101
    0.0926
   -0.0310
   -0.1526
   -0.2641
   -0.3584
   -0.4293
   -0.4720
   -0.4839
   -0.4642
   -0.4142
   -0.3375
   -0.2400
   -0.1316
   -0.0329

Number of Iterations required:
    21

Tolerance:
   1.0000e-10

MATLAB built-in solution:
    0.0329
    0.1316
    0.2400
    0.3375
    0.4142
    0.4642
    0.4839
    0.4720
    0.4293
    0.3584
    0.2641
    0.1526
    0.0310
   -0.0926
   -0.2101
   -0.3138
   -0.3971
   -0.4544
   -0.4819
   -0.4780
   -0.4427
   -0.3785
   -0.2896
   -0.1817
   -0.0619
    0.0619
    0.1817
    0.2896
    0.3785
    0.4427
    0.4780
    0.4819
    0.4544
    0.3971
    0.3138
    0.2101
    0.0926
```

```
-0.0310
-0.1526
-0.2641
-0.3584
-0.4293
-0.4720
-0.4839
-0.4642
-0.4142
-0.3375
-0.2400
-0.1316
-0.0329
```

2 C)

```
i=1;
for w=0.38:0.025:1.73
    [xit,iter]=SoR(x0,Ait,bit,tol,false,w);
    Iteration(i)=iter;
    Omega(i)=w;
    i=i+1;
end
[Min, Index] = min(Iteration);
disp('Relaxation parameter (Omega) that minimizes iterations: ')
disp(Omega(Index));
disp('Number of iterations performed with that Relaxation parameter (Omega): ')
disp(Iteration(Index));
```

```
Warning: Solution may not have converged fully...
Warning: Solution may not have converged fully...
Relaxation parameter (Omega) that minimizes iterations:
    1.1050

Number of iterations performed with that Relaxation parameter (Omega):
    21
```

2 D)

```
[xit,iter]=SoR(x0,Ait,bit,tol,false,1);
disp('Number of iterations performed by Gauss-Seidel: ')
disp(iter);
disp('Difference in number of iterations performed by Gauss-Seidel vs Optimal case: ')
disp(abs(iter-Iteration(Index)));
j=1;
for i=1:length(Iteration)
    if iter==Iteration(i)
        ind(j,1)=i;
        ind(j,2)=Omega(i);
        j=j+1;
    end %if
end %for
disp('The lowest value of the Relaxation parameter (Omega) that performs atleast as well as the Gauss-Seidel method: ')
disp(ind(1,2));
disp('The highest value of the Relaxation parameter (Omega) that performs atleast as well as the Gauss-Seidel method: ')
disp(ind(2,2));
```

```
Number of iterations performed by Gauss-Seidel:
    26
```

```
Difference in number of iterations performed by Gauss-Seidel vs Optimal case:
     5

The lowest value of the Relaxation parameter (Omega) that performs atleast as well as the Gauss-Seidel method:
    1.0050

The highest value of the Relaxation parameter (Omega) that performs atleast as well as the Gauss-Seidel method:
    1.2300
```