Table of Contents


```
clear all
close all
clc
%loading data for problem
load C:\Users\Vittorio\EP501_assignments\assignments
\HW2\iterative_testproblem.mat
%initial guess vector
x0=zeros(size(Ait,1),1);
%vector of omegas, in order to get the best one for the problem
%it was observed that solution could not converge for an omega over
 around
%1.35, and either way, iteration had an increasing trend
omega=linspace(1.01,1.32,80);
```

Part a) and b)

```
%for loop to calculate the SOR method solution for each omega
for k=1:length(omega)
    [x(:,k),nit(k)]=SOR_method(x0,Ait,bit,1e-8,omega(k));
end

%checking results with the built-in MatLab function
xcheck=Ait\bit;

%outputs
disp('SOR method solution:')
disp(x(:,1))
disp('Built-in MatLab solution:')
disp(xcheck)
```

```
SOR method 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

Built-in MatLab 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
```

Part c)

```
%minimum number of iteration needed
iter=min(nit);

%extracting the best omega, knowing the position of the minimum number
 of
%iterations used in the iteration array
for i=1:length(nit)
    if iter==nit(i)
        bestomega=omega(i);
        break
    end
end
%outputs
```

```
fprintf('The optimal value for omega is = %1.4f\n', bestomega)
```

```
The optimal value for omega is = 1.0335
```

Part d)

```
%Gauss-Seidel iteration (omega=1)
[xgauss,nitgauss]=SOR_method(x0,Ait,bit,1e-8,1);
diff=nitgauss-iter;
%checking which omegas do worse than Gauss-Seidel in terms of
 iterations
%needed
for i=1:length(nit)
    if nit(i)>nitgauss
        fprintf('Omega values over %1.4f do worse than the Gauss-
Seidel method!\n',omega(i))
        fprintf('The SOR method takes %i iterations less than the
 Gauss-Seidel method\n',diff)
        break
    end
end
```

```
Omega values over 1.0728 do worse than the Gauss-Seidel method!
The SOR method takes 2 iterations less than the Gauss-Seidel method
```

Published with MATLAB® R2020a