Math 105AL Final Project

Kian Kermani

ID: 11586591

```
%Initializing our variables

n = 20; % dimension number
h = pi/(n + 1); % dividing up our interval [0,pi]
x = zeros([1,n+1]); % initialize x-val array
for i = 1:n+1
    x(i+1) = i*h;
end
x = x(2:21); % getting rid of boundary points
A = zeros(n); % initialize coeff. matrix
for i = 1:n
    A(i,i) = -2;
        if i ~= n
            A(i,i+1) = 1;
            A(i+1,i) = 1;
        end
end

f = sin(x) * h^2;

%Define T_j matrix below
D = diag(diag(A));
L = tril(A,-1);
U = triu(A,1);

T_j = -inv(D) * (L + U);
```

Problem 1

We will use three different numerical methods to solve the system Au = f.

```
my_jacobi(n,A,f,zeros([1,n]),1e-4,1000);
```

```
Jacobi: Our approximate solution is given by
   -0.1489
   -0.2945
   -0.4335
   -0.5628
   -0.6796
   -0.7812
   -0.8653
   -0.9301
   -0.9741
   -0.9964
   -0.9964
   -0.9741
   -0.9301
```

```
   -0.8653
   -0.7812
   -0.6796
   -0.5628
   -0.4335
   -0.2945
   -0.1489

Jacobi number of iterations: 526
```

```
my_siedel(n,A,f,zeros([1,n]),1e-4,1000);
```

```
Gauss: Our approximate solution is given by
   -0.1491
   -0.2949
   -0.4341
   -0.5636
   -0.6805
   -0.7822
   -0.8664
   -0.9313
   -0.9754
   -0.9977
   -0.9977
   -0.9755
   -0.9314
   -0.8665
   -0.7823
   -0.6806
   -0.5637
   -0.4342
   -0.2949
   -0.1491

Gauss-Siedel number of iterations: 295
```

```
u_approx_sor = my_SOR(n,A,f,zeros([1,n]),1.5,1e-4,1000);
```

```
SOR: Our approximate solution is given by
   -0.1492
   -0.2952
   -0.4345
   -0.5641
   -0.6811
   -0.7829
   -0.8673
   -0.9322
   -0.9764
   -0.9987
   -0.9987
   -0.9764
   -0.9323
   -0.8673
   -0.7830
   -0.6812
   -0.5642
   -0.4346
   -0.2952
```

```
   -0.1493

SOR number of iterations: 112
```

We can verify that we have approximated the correct u by the following command.

```
exact_sol = linsolve(A,transpose(f))
```

```
exact_sol = 20×1
   -0.1493
   -0.2953
   -0.4347
   -0.5644
   -0.6814
   -0.7833
   -0.8676
   -0.9326
   -0.9767
   -0.9991
      ⋮
```

Our approximations seem to agree with that result, so we may now direct our focus toward analyzing which method converged fastest.

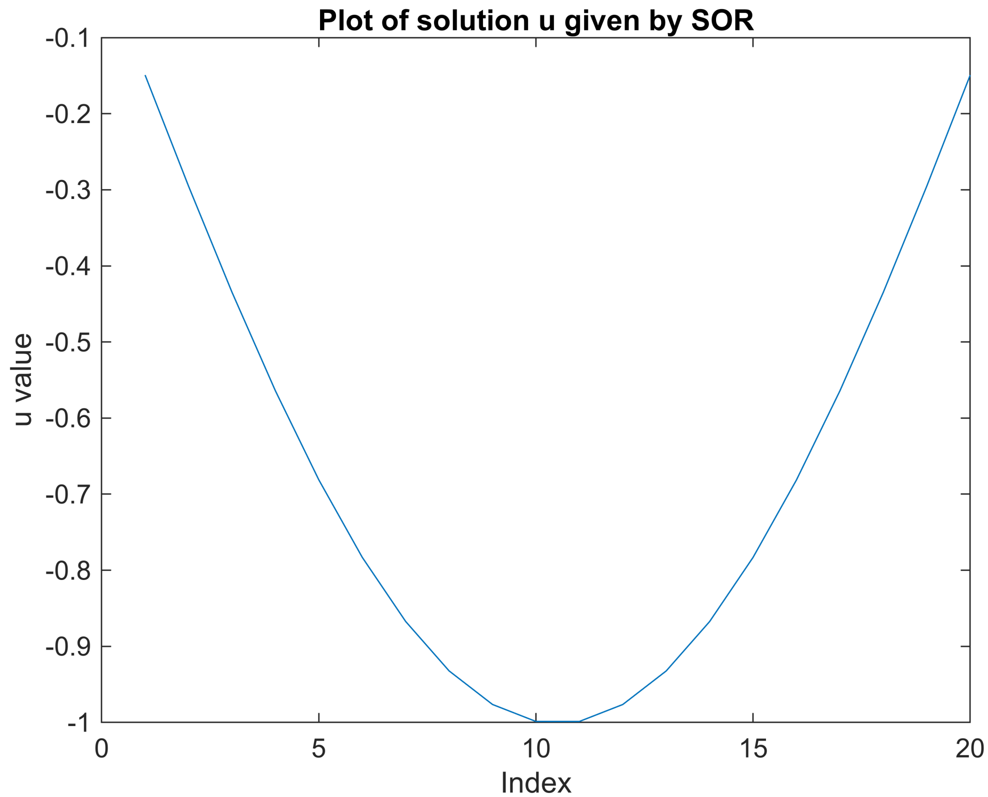
Provlem 2

Based on our results from above, we see that the SOR method has the fastest convergence, with 112 iterations required to get an approximate solution.

Problem 3

We now plot the approximate solution given by the SOR method where the x-axis represents the index and y-axis represents the value of u.

```
figure
plot(u_approx_sor)
title('Plot of solution u given by SOR')
xlabel('Index')
ylabel('u value')
```



Problem 4

Lastly, we implement the power method to get the largest eigenvalue of our already defined T_j matrix.

```
power_method(n,T_j,ones([n,1]),1e-4,1000)
```

```
Mu = 9.888175e-01
    0.1495
    0.2956
    0.4351
    0.5649
    0.6821
    0.7840
    0.8685
    0.9335
    0.9777
    1.0000
    1.0000
    0.9777
    0.9335
    0.8685
    0.7840
    0.6821
    0.5649
    0.4351
    0.2956
    0.1495

Power method number of iterations: 162
```

We can verify this is correct using the following command.

```
max(eig(T_j))
```

```
ans = 0.9888
```

Which is very close to our Mu from the power method, hence we have succeeded.

Defined functions

```
function output = my_jacobi(n,A,b,XO,TOL,N)
    k = 1;
    x = zeros([1,n]);
    while k <= N
        for i = 1:n
            index_arr = 1:n;
            index_arr(i) = [];
            x(i) = (b(i) - sum( A(i,index_arr) .* XO(index_arr)) ) / A(i,i);
        end

        if norm(x - XO) < TOL
            disp('Jacobi: Our approximate solution is given by');
            disp(transpose(x));
            fprintf('Jacobi number of iterations: %d \n',k);
            return
        end

        k = k + 1;

        for i = 1:n
            XO(i) = x(i);
        end

    end
    fprintf('Max number of iterations exceeded \n');
    return
end

function output = my_siedel(n,A,b,XO,TOL,N)
    k = 1;
    x = zeros([1,n]);
    while k <= N
        for i = 1:n
            x(i) = (b(i) - sum(A(i,1:i-1) .* x(1:i-1) ) - sum( A(i,i+1:n) .*
XO(i+1:n) ) ) / A(i,i);
        end

        if norm(x - XO) < TOL
            disp('Gauss: Our approximate solution is given by');
            disp(transpose(x));
            fprintf('Gauss-Siedel number of iterations: %d \n',k);
            return
```

```
        end

        k = k + 1;

        for i = 1:n
            XO(i) = x(i);
        end

    end
    fprintf('Max number of iterations exceeded \n');
end

function output = my_SOR(n,A,b,XO,omega,TOL,N)
    k = 1;
    x = zeros([1,n]);
    while k <= N
        for i = 1:n
            x(i) = (1 - omega)*XO(i) + ( omega/A(i,i) )*(b(i) -
sum( A(i,1:i-1).*x(1:i-1) ) - sum( A(i,i+1:n).*XO(i+1:n) ) );
        end

        if norm(x - XO) < TOL
            disp('SOR: Our approximate solution is given by');
            disp(transpose(x));
            fprintf('SOR number of iterations: %d \n',k);
            output = x;
            return
        end

        k = k + 1;
        for i = 1:n
            XO(i) = x(i);
        end

    end

    fprintf('Max number of iterations exceeded \n');
end

function output = power_method(n,A,x,TOL,N)
    k = 1;
    for i = 1:n
        if abs(x(i)) == max(abs(x))
            p = i;
            break
        end
    end
    x = x ./ x(p);

    while k <= N
```

```
        y = mtimes(A,x);
        mu = y(p);
        for i = 1:n
            if abs(y(i)) == max(abs(y))
                p = i;
                break
            end
        end
        if y(p) == 0
            disp('Eigenvector')
            disp(x)
            disp('A has the eigenvalue 0, select a new vector x and restart')
            return
        end

        ERR = max(abs(x - (y ./ y(p) ) ) );
        x = y./y(p);
        if ERR < TOL
            fprintf('Mu = %d \n',mu);
            disp(x);
            fprintf('Power method number of iterations: %d \n',k);
            return
        end

        k = k + 1;
    end

    disp('The maximum number of iterations has been exceeded')
end
```