Homework 3

AE370 - Spring 2018

Emilio R. Gordon

Problem: For the systems of equations

$$\begin{aligned} 5x_1 - 15x_2 - x_3 + 2x_4 &= 11 \\ 3x_1 + 20x_2 + 4x_3 &= -8 \\ x_1 + 6x_2 + 11x_3 - 3x_4 &= 7 \\ 4x_1 + 11x_2 + 3x_3 + 12x_4 &= 4 \end{aligned}$$

Solve the equations using the Jacobi method and the Gauss-seidel method. Stop when the system residual is of the same order of magnitude as the corresponding "exact solution" from Matlab's operator

Solution:

The Jacobi Method converges within a tolerance of 1e-14 after the 74 iteration with a final solution:

$$u_{\text{Jacobi}} = \begin{bmatrix} 0.215948489351164 \\ -0.659422981674096 \\ 1.135153541357108 \\ 0.582033184744923 \end{bmatrix}$$

The Gauss-Seidel Method converges within a tolerance of 1e-14 after the 32 iteration with a final solution:

$$u_{\text{Gauss-Seidel}} = \begin{bmatrix} 0.215948489351164 \\ -0.659422981674096 \\ 1.135153541357108 \\ 0.582033184744924 \end{bmatrix}$$

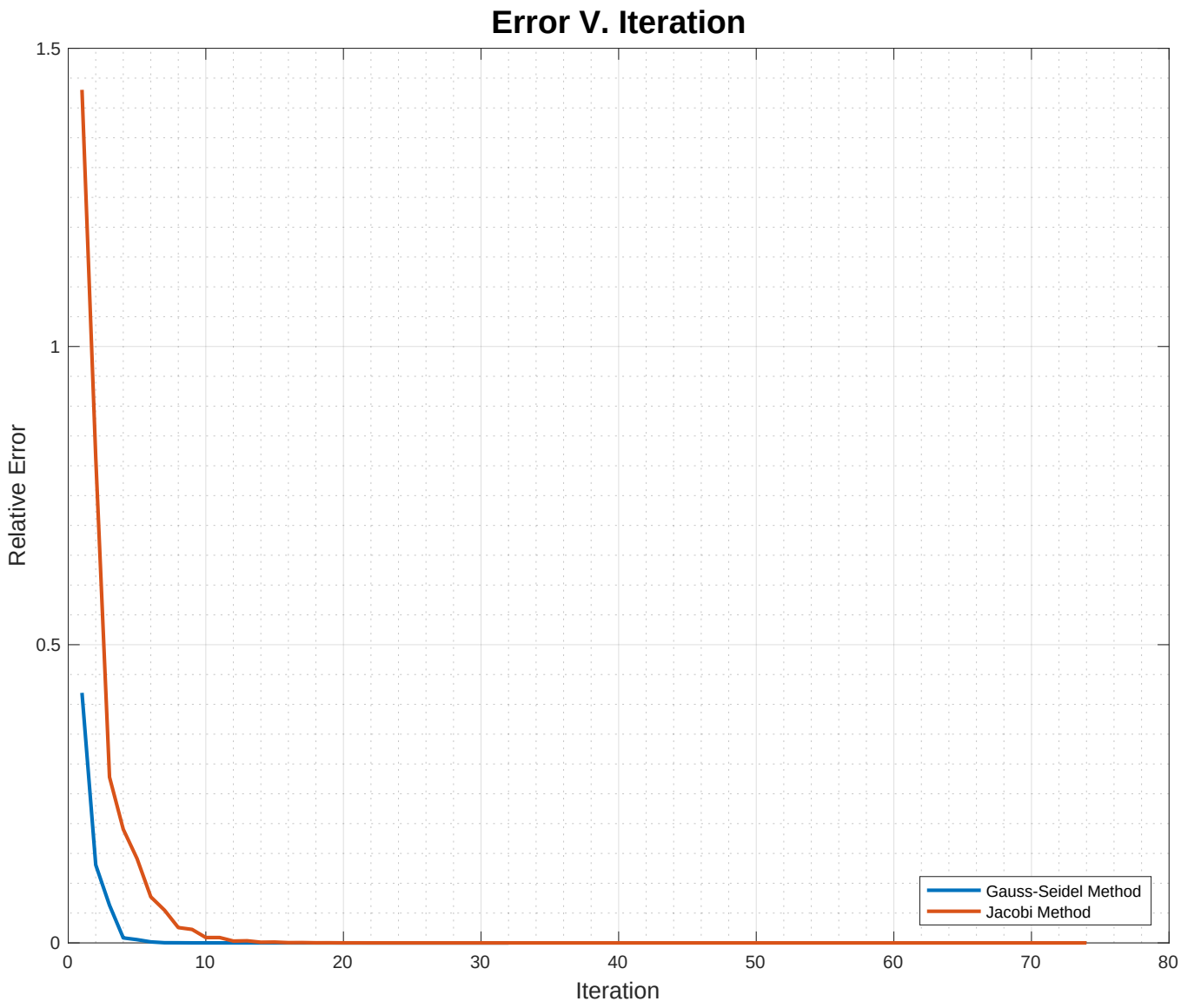
The Matlab Function A\B = u has the final solution of:

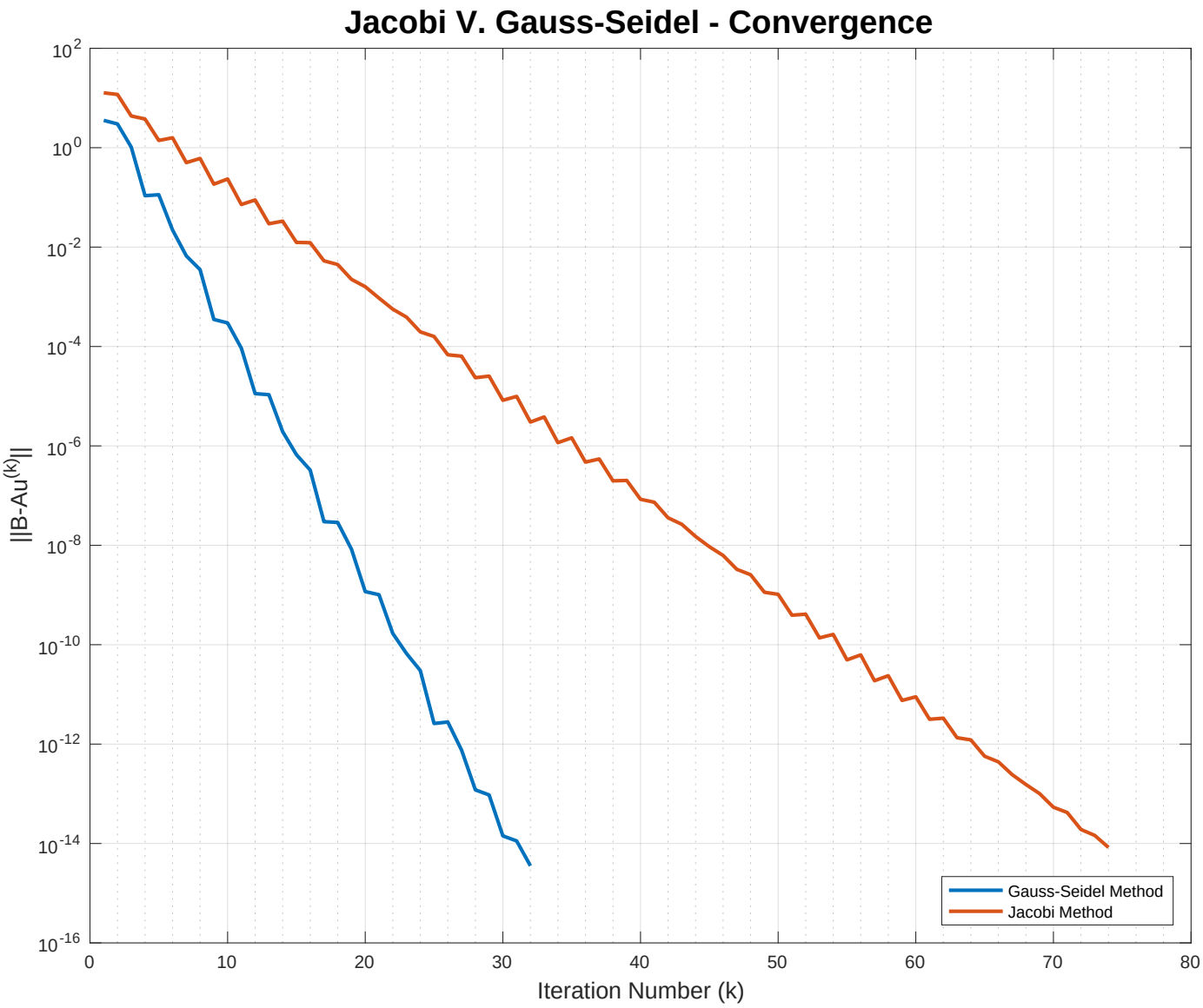
$$u_{\text{MatlabExact}} = \begin{bmatrix} 0.215948489351164 \\ -0.659422981674096 \\ 1.135153541357107 \\ 0.582033184744923 \end{bmatrix}$$

As we can see, both methods converged to the Matlab "exact" value. Note that the Gauss-Seidel method converges in half the number of iterations needed for the Jacobi method.

The following code was utilized to arrive at these solutions. Matlab can produce up to 15 decimal places. As a result, tolerance was set to 1e-14 to reach an order of magnitude difference of one.

```
function hw3
    tol = 1e-14;                                    % Tolerance
    A = [5 -15 -1 2; 3 20 4 0; 1 6 11 -3;4 11 3 12];% A Matrix defined by problem statement
    B = [11; -8; 7; 4];                             % B Matrix defined by problem statement
    exact = A\B;                                    % Exact Matlab Solution
    [uJ iterationsJ] = jacobi(A,B,tol);             % Run Jacobi Function
    [uGS iterationsGS] = gaussSeidel(A,B,tol);      % Run Gauss-Seidel Function
    plotter(A,B,tol)                                % Run Plotter Function
end
function [uJ iterations] = jacobi(A,B,tolerance)
    u = [0;0;0;0];
    M = zeros(size(A));
    N = -A;                  % Negative A
    for i=1:size(A,1)
        M(i,i) = A(i,i);     % Diagonal A
        N(i,i) = 0;          % Zero Diagonal Components
    end
    i = 0;
    while norm(B-A*u)>tolerance
        i=i+1;
        u = inv(M)*(B+N*u); % u_k+1 = M^-1 [B+N*u^k]
        iterations(i) = i;
        uJ(1:4,i) = u;
    end
end
function [uGS iterations] = gaussSeidel(A,B,tolerance)
    u = [0;0;0;0];
    M = triu(A);       % M is the Triangular Upper Bound of A
    N = M-A;           % N is the Triangular Lower Bound of A
    i = 0;
    while norm(B-A*u)>tolerance
        i=i+1;
        u = inv(M)*(B+N*u); % u_k+1 = M^-1 [B+N*u^k]
        iterations(i) = i;
        uGS(1:4,i) = u;
    end
end
function plotter(A,B,tol)
    [uJ iterationsJ] = jacobi(A,B,tol);
    [uGS iterationsGS] = gaussSeidel(A,B,tol);
    for j=1:length(uGS)
        GSRE(j) = norm(uGS(:,j)-(A\B))/norm(A\B);  % Compute Relative Error
        GSresidue(j) = norm(B-A*uGS(:,j));         % Compute Residue
    end
    for k=1:length(uJ)
        JRE(k) = norm(uJ(:,k)-(A\B))/norm(A\B);       % Compute Relative Error
        Jresidue(k) = norm(B-A*uJ(:,k));              % Compute Residue
    end
    plot(iterationsGS, GSRE,'LineWidth',2)
    hold
    plot(iterationsJ, JRE,'LineWidth',2)
    semilogy(iterationsGS,GSresidue,iterationsJ,Jresidue,'linewidth',2)
    %Significant Plotting Code removed for space
end
```


Error V. Iteration
Relative Error
Iteration
1.5
1
0.5
0
0
10
20
30
40
50
60
70
80
Gauss-Seidel Method
Jacobi Method


Jacobi V. Gauss-Seidel - Convergence
||B-Au(k)||
Iteration Number (k)
Gauss-Seidel Method
Jacobi Method
10^2
10^0
10^-2
10^-4
10^-6
10^-8
10^-10
10^-12
10^-14
10^-16
0
10
20
30
40
50
60
70
80