Problem I:

Using problem 4.6 in the attached pages from Larson (2007)

(a) Apply full pivoting to this matrix to improve diagonal dominancy and check to see if the matrix is diagonally dominant or not before and after pivoting.

(b) Find the condition number before and after scaling the equations.

(c) Find the determinant of the scaled matrix using the information from parts (a), (b) and (c)

(d) Comment on the condition of the scaled matrix.

(e) Solve Problem 4.6 using the following methods and comment on the pros and cons of each method when applied to this problem.
   - Gaussian Elimination
   - Gauss-Jordan Elimination
   - Jacobi Iteration
   - L-U Decomposition
   - Gauss-Siedel Iteration
a) The system of equations in matrix form comes to:
\[
\begin{bmatrix}
-0.0752 & 0.08 & -0.0088 \\
-0.01651 & 0.02 & -0.0022 \\
0 & 0.9 & -0.9890
\end{bmatrix}
\begin{bmatrix}
S_2 \\
S_3 \\
S_4
\end{bmatrix} =
\begin{bmatrix}
-1 \\
0.0 \\
0.0
\end{bmatrix}
\]
After scaling to simplify pivoting,
\[
\begin{bmatrix}
-0.94025 & 1.0 & -0.11 \\
-0.8255 & 1.0 & -0.11 \\
0 & -0.91 & 1.0
\end{bmatrix}
\begin{bmatrix}
S_2 \\
S_3 \\
S_4
\end{bmatrix} =
\begin{bmatrix}
-12.5 \\
0.0 \\
0.0
\end{bmatrix}
\]
In no way is it possible to fully pivot the matrix into a diagonally dominant form.

b) The inverse of A comes to:
\[
\begin{bmatrix}
-108.9325 & 435.7298 & 0.0 \\
-99.9265 & 455.2678 & -0.1236 \\
-90.9341 & 414.2983 & -1.1236
\end{bmatrix}
\]
before scaling.
Once it is scaled, A’s inverse is:
\[
\begin{bmatrix}
-8.7146 & 8.7146 & 0.0 \\
-7.9941 & 9.1053 & 0.1222 \\
-7.2746 & 8.2859 & 1.1112
\end{bmatrix}
\]
Finding the Frobenius norm for each A and its inverse,

Original: $||A|| = 1.3420$ Matlab provides an answer of 1.3387 $||A^{-1}|| = 773.8826$ Matlab provides an answer of 773.8098

Scaled: $||A|| = 2.3276$ Matlab provides an answer of 2.0966 $||A^{-1}|| = 20.5311$ Matlab provides an answer of 20.5017

c) Thusly, the condition prior to scaling is 1038.55 (1035.9 via Matlab). After scaling, it comes to 47.79 (42.98 via Matlab). As one can see, the condition improves vastly once the matrix is scaled.

d) Using MATLAB to calculate the determinant, its value was found to be -0.1033.
e) Because the condition of the original matrix is so askew, only the scaled matrix will be used in the elimination procedures.

Gaussian Elimination:
S_2 = 108.9325
S_3 = 99.9264
S_4 = 90.9330
This method simplifies one equation down to one variable so that it is very easy to solve, then allows backward substitution to solve for the others. The only downside is the tedious backward substitution involved.

Gauss-Jordan Elimination:
S_2 = 108.9325
S_3 = 99.9264
S_4 = 90.9330
This method solves for each variable individually and therefore eliminates the backward substitution. The method itself, however, is almost as tedious as the substitution.

Jacobi Iteration:
S_2 = 47.4140
S_3 = 153.8667
S_4 = 42.9133
After 100 iterations of the Jacobi method, it is clear that it is diverging, which is a rather big problem to using that particular method. If it converges, however, it converges quickly and precisely.

L-U Decomposition:
S_2 = 108.9325
S_3 = 99.9264
S_4 = 90.9330
L-U Decomposition uses a complicated method to come up with the matrices one uses in order to find the solutions to the equations. Despite the arduous task of finding the U and L matrices, it is quite useful for large amounts of equations and is widely used for its efficiency.

Gauss-Seidel Iteration:
S_2 = 108.9325
S_3 = 99.9264
S_4 = 90.9330
In many ways similar to the Jacobi method, the Gauss-Seidel method tends to converge very quickly and accurately where Jacobi method does not.
Matlab code

clear
clc
%Define Matrices
a = [-0.07522 0.08 -0.0088
    -0.01651 0.02 -0.0022
    0 0.9 -0.9890];

b = [-.94025 1.0 -0.11
     -0.8255 1.0 -0.11
     0 -0.91 1.0];

aminus = inv(a);
bminus = inv(b);

%c-f are the Frobenius norms

c = 0;
for i = 1:3
    for j = 1:3
        c = c + (a(i, j))^2;
    end
end
c = c^0.5;

d = 0;
for i = 1:3
    for j = 1:3
        d = d + (b(i, j))^2;
    end
end
d = d^0.5;

e = 0;
for i = 1:3
    for j = 1:3
        e = e + (aminus(i, j))^2;
    end
end
e = e^0.5;

f = 0;
for i = 1:3
    for j = 1:3
        f = f + (bminus(i, j))^2;
    end
end
f = f^0.5;

%Find determinant

det = 0;
for i = 1:3
    if i == 1
        inside = b(2,2)*b(3,3)-b(3,2)*b(2,3);
    end
    if i == 2
        outside = b(1,2)*b(3,3)-b(3,2)*b(1,3);
    end
    if i == 3
        outside = b(1,2)*b(2,3)-b(2,2)*b(1,3);

    det = det + inside + outside;
end

%Print results

class = ['frobenius norm of a = ', num2str(c), ' f of a = ', num2str(d), ' f of a = ', num2str(e), ' f of a = ', num2str(f), ' determinant = ', num2str(det)];
for i = 1:length(class)
    if i == 1
        fprintf('%s
', class{i});
    else
        fprintf(', %s', class{i});
    end
end
inside = b(2,1)*b(3,3)-b(2,3)*b(3,1);
end
if i == 3
    inside = b(2,1)*b(3,2)-b(2,3)*b(3,1);
end
det = det + (-1)^(i+1)*b(1,i)*inside;
end

%Guassian Elimination
b = [-.94025   1.0  -0.11  -12.5
     -0.8255   1.0  -0.11  0.0
     0         -0.91  1.0  0.0];
row1 = b(1, 1:4);
row2 = b(2, 1:4);
row3 = b(3, 1:4);
%Change first column
row2 = row2 - row1*b(2, 1)/b(1, 1);
row3 = row3 - row1*b(3, 1)/b(1, 1);
b = [row1
     row2
     row3];
%Change second column
row3 = row3 - row2*b(3, 2)/b(2, 2);
b = [row1
     row2
     row3];
S4 = b(3, 4)/b(3,3);
S3 = (b(2, 4)-S4*b(2, 3))/b(2, 2);
S2 = (b(1, 4)-S4*b(1, 3)-S3*b(1, 2))/b(1, 1);

%Gauss-Jordan Elimination
b = [-.94025   1.0  -0.11  -12.5
     -0.8255   1.0  -0.11  0.0
     0         -0.91  1.0  0.0];
row1 = b(1, 1:4);
row2 = b(2, 1:4);
row3 = b(3, 1:4);
%Change first column
row1 = row1/b(1,1);
b = [row1
     row2
     row3];
row2 = row2 - row1*b(2, 1)/b(1, 1);
row3 = row3 - row1*b(3, 1)/b(1, 1);
b = [row1
     row2
     row3];
%Change second column
row2 = row2/b(2, 2);
b = [row1
     row2
     row3];
row1 = row1 - row2*b(1, 2)/b(2, 2);
row3 = row3 - row2*b(3, 2)/b(2, 2);
% Change third column
row3 = row3/b(3, 3);

row2 = row2 - row3*b(2, 3)/b(3, 3);
row1 = row1 - row3*b(1, 3)/b(3, 3);

b = [row1
     row2
     row3];

S4 = b(3, 4);
S3 = b(2, 4);
S2 = b(1, 4);

% Jacobi Iteration
S2 = 100;
S3 = 100;
S4 = 100;
for i = 1:100
    S2new = (12.5-(-.11*%4)-S3)/-0.94025;
    S3new = 0.8255*S2+0.11*S4;
    S4new = 0.91*S3;
    S2 = S2new;
    S3 = S3new;
    S4 = S4new;
end

% L-U Decomposition
b = [-0.94025   1.0   -0.11
     -0.8255   1.0   -0.11
        0   -0.91   1.0];
l = [0 0 0
     0 0 0
     0 0 0];
u = [1 0 0
     0 1 0
     0 0 1];
for i = 1:3
    l(i, 1) = b(i, 1);
end
u(1,2) = b(1,2)/l(1,1);
u(1,3) = b(1,3)/l(1,1);
l(2,2) = b(2,2)-l(2,1)*u(1,2);
l(3,2) = b(3,2)-l(3,1)*u(1,2);
u(2,3) = (b(2,3)-l(2,1)*u(1,3))/l(2,2);
l(3,3) = b(3,3)-l(3,1)*u(1,3)-l(3,2)*u(2,3);
q1 = -12.5/l(1,1);
q2 = -q1*l(2,1)/l(2,2);
q3 = -q2*l(3,2)-q1*l(3,1))/l(3,3);
S4 = q3;
S3 = q2-S4*u(2,3);
S2 = q1-S3*u(1,2)-S4*u(1,3);
% Gauss-Seidel Iteration
S2 = 100;
S3 = 100;
S4 = 100;
for i = 1:100
    S2 = (-12.5 - (-.11*S4) - S3) / -0.94025;
    S3 = 0.8255*S2 + 0.11*S4;
    S4 = 0.91*S3;
end
S2;
S3;
S4;