TO: Professor Andreas Linninger / Grant Hartung

FROM: Kehinde Abioye

DATE: September 4, 2018

SUBJECT: Iterative methods for solving linear algebraic equations 2

The purpose of this assignment is to use the Gauss-Seidel, Jacobi, and Gradient method to solve for systems of linear equations.

Introduction

Part one uses different iterative solves to solve for three set of equations. Iterative methods are used when Gauss elimination can't be used or less efficient. The iterative methods used were Gradient, Gauss-Seidel, and Jacobi method. Part two involves visualizing linesearch to display a 2D section of the curve.

Methods

Part 1

Part a:

Equations 1 and 2 were solved for using Gauss elimination.

$$A = \begin{bmatrix} 2 & 3 \\ 1 & 4 \end{bmatrix}; b = \begin{bmatrix} 8 \\ 9 \end{bmatrix} \quad (1)$$

$$A = \begin{bmatrix} 2 & -1 \\ -1 & 3 \end{bmatrix}; b = \begin{bmatrix} 3 \\ 11 \end{bmatrix} \quad (2)$$

The matrices were reduced to row reduced echelon form

Part 1b:

A contour map of the residual surface was plotted in MATLAB for the first and second equations using the command contour. Ranges for x and y were set to -10 to 10. A for loop was created for the matrices in residual form. Then the residual surface was calculated using equation 4:

$$r = (Ax - b) \quad (3)$$

$$\varphi(x) = r^T r = (Ax - b)^T (Ax - b) \quad (4)$$

The surf command was used.

Part c:

The direction of steepest ascent and steepest descent was drawn by hand from 5 random points on the contour plot. The optimal acceleration along the steepest descent direction was also plotted.

Part d:

Gradient method and Gauss-Seidel method was used for equation 1 and 2, and the convergence history was plotted on the contour maps. Three initial guesses were used: x= [1;1;1], x=[2;2;2], and x=[3;3;3].

Part e:

Gauss elimination was used to solve for equation 5.

$$A = \begin{bmatrix} 4 & 2 & 1 \\ 2 & 3 & 1 \\ 1 & -1 & 3 \end{bmatrix}; b = \begin{bmatrix} 18 \\ 17 \\ 7 \end{bmatrix} \qquad (5)$$

Part f:

Gauss-Seidel and Jacobi methods were used to solve for equation 5. The same three initial guesses from part 4 was used.

Part 2

Part a:

Linesearch was performed with a search direction of $[-12\ -5]^T$. The initial guess was x = [10;10].

Part b:

Minimization was performed using

$$\min\alpha\ [A(x+\alpha p)-b]^T[A(x+\alpha p)-b] \qquad (6)$$

Part c:

The minimization was solved symbolically by using equation 7 to verify the solution in part b.

$$a = \frac{r^T A p}{p^T A^T A p}$$

Results

Part 1a:

The solutions using gauss elimination for equations one and two respectively:

x = [1; 2]

x = [4; 5]

Part 1b:

Contour plot of residual error for equation 1:

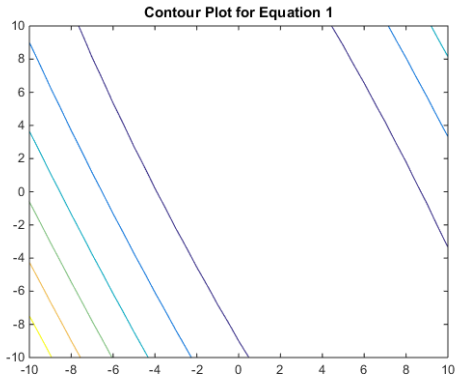

Figure 1 contour plot for first equation

Surface plot of residual error for equation 1:

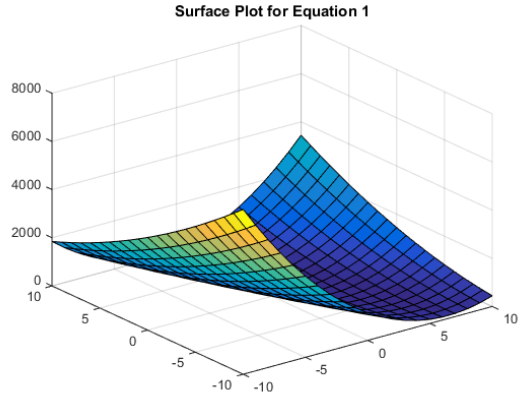

Figure 2 Surface plot for first equation

Contour plot of residual error for equation 2:

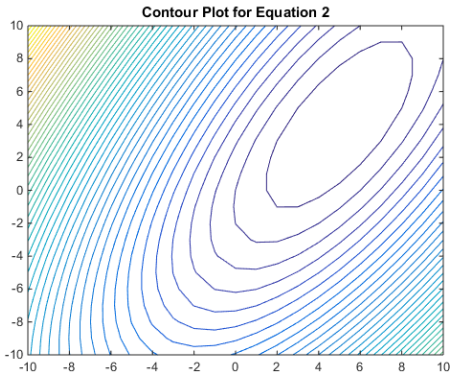

Figure 3 Contour plot for second equation

Surface plot of residual error for equation 2:

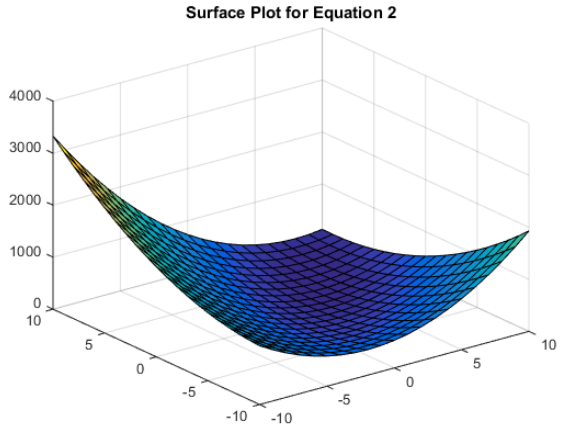

Figure 4 Surface plot for second equation

Part 2

Part a:

Linesearch to find the optimal value of acceleration is shown in figure 5:

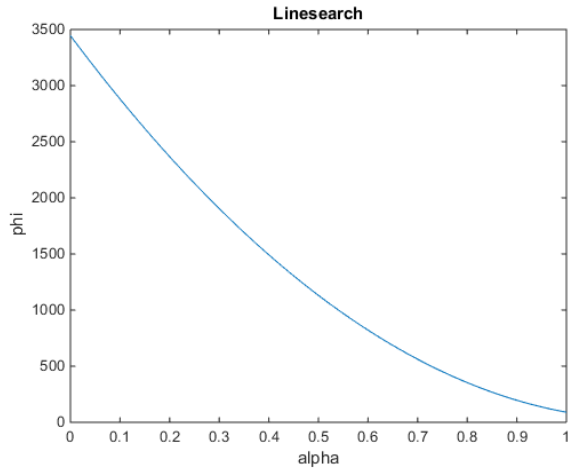

Figure 5 linesearch for equation 1

Conclusion

Different iterative methods were used to solve for the equations. The same solution resulted for each equation. Linesearch used to plot a 2D plot of the contour plot. This helps to see the solution easily.

Discussion

Iterative methods were studied more in this assignment because it is a useful tool when solving for system of equations. Real world set of equations will be more extensive and this is where iterative methods come into play.

Appendix A

```
clc
clear all
close all

% function xUpdate = GradientMethod(x,b,A)
% tol = 1e-6;
% xUpdate(:,1) = x; %initial guess is first iteration
% r = b-A*x; %Compute first residual
% for i = 2:100 %start from second iteration
%  phi=(r')*r;
%  alpha = phi/(r'*A*r);
%  xUpdate(:,i) = x + alpha*r;
%  rNew = r - alpha*A*r;
%  Euclidean=sqrt((rNew(1)^2)+(rNew(2)^2)); %measures length of rNew vctor
%  if Euclidean < tol %if the magnitude of the vector is less than the
tolerance
%                     %then break from the for loop and print solution x
%  xUpdate(:,i)
%  break %breaks from for loop
%  end
%  x = xUpdate(:,i);%update x to reiterate with new x
%  r = rNew; %update r to reiterate with new r
% end
% end
%
A1=[2 3; 1 4]; b1=[8;9];
A2=[2 -1;-1 3]; b2=[3;11];
A3=[4 2 1;2 3 1;1 -1 3]; b3=[18;17;7];
Ab1=[A1 b1];
Ab2=[A2 b2];
Ab3=[A3 b3];
x=[1;1;1]
% x=[2;2;2]
% x=[3;3;3]

% GradientMethod(x,b1,A1)
% GradientMethod(x,b2,A2)

%Gauss-Siedel Method for 1
% x1=(b1(1)-(A1(2)*x(2)))/A1(1);
% x2=(b1(2)-(A1(3)*x1))/A1(4);
% x=[x1;x2];
% x1=(b1(1)-(A1(2)*x(2)))/A1(1);
% x2=(b1(2)-(A1(3)*x1))/A1(4);
% x=[x1;x2];
% x1=(b1(1)-(A1(2)*x(2)))/A1(1);
% x2=(b1(2)-(A1(3)*x1))/A1(4);
% x=[x1;x2];
% x1=(b1(1)-(A1(2)*x(2)))/A1(1);
% x2=(b1(2)-(A1(3)*x1))/A1(4);
% x=[x1;x2]
% x1=(b1(1)-(A1(2)*x(2)))/A1(1);
% x2=(b1(2)-(A1(3)*x1))/A1(4);
% x=[x1;x2]
```

```
% x1=(b1(1)-(A1(2)*x(2)))/A1(1);
% x2=(b1(2)-(A1(3)*x1))/A1(4);
% x=[x1;x2]

%Gauss-Siedel Method for 2
% x1=(b2(1)-(A2(1,2)*x(2)))/A2(1,1);
% x2=(b2(2)-(A2(2,1)*x1))/A2(2,2);
% x=[x1;x2];
% x1=(b2(1)-(A2(2)*x(2)))/A2(1);
% x2=(b2(2)-(A2(3)*x1))/A2(4);
% x=[x1;x2];
% x1=(b2(1)-(A2(2)*x(2)))/A2(1);
% x2=(b2(2)-(A2(3)*x1))/A2(4);
% x=[x1;x2];
% x1=(b2(1)-(A2(2)*x(2)))/A2(1);
% x2=(b2(2)-(A2(3)*x1))/A2(4);
% x=[x1;x2];
% x1=(b2(1)-(A2(2)*x(2)))/A2(1);
% x2=(b2(2)-(A2(3)*x1))/A2(4);
% x=[x1;x2];
% x1=(b2(1)-(A2(2)*x(2)))/A2(1);
% x2=(b2(2)-(A2(3)*x1))/A2(4);
% x=[x1;x2];
% x1=(b2(1)-(A2(2)*x(2)))/A2(1);
% x2=(b2(2)-(A2(3)*x1))/A2(4);
% x=[x1;x2];
% x1=(b2(1)-(A2(2)*x(2)))/A2(1);
% x2=(b2(2)-(A2(3)*x1))/A2(4);
% x=[x1;x2];
% x1=(b2(1)-(A2(2)*x(2)))/A2(1);
% x2=(b2(2)-(A2(3)*x1))/A2(4);
% x=[x1;x2]


% %Gauss-Siedel Method for 3
% x1=(b3(1)-(A3(1,2)*x(2))-(A3(1,3)*x(3)))/A3(1,1);
% x2=(b3(2)-(A3(2,1)*x1)-(A3(2,3)*x(3)))/A3(2,2);
% x3=(b3(3)-(A3(3,1)*x1)-(A3(3,2)*x2))/A3(3,3);
% x=[x1;x2;x3]
% x1=(b3(1)-(A3(1,2)*x(2))-(A3(1,3)*x(3)))/A3(1,1);
% x2=(b3(2)-(A3(2,1)*x1)-(A3(2,3)*x(3)))/A3(2,2);
% x3=(b3(3)-(A3(3,1)*x1)-(A3(3,2)*x2))/A3(3,3);
% x=[x1;x2;x3];
% x1=(b3(1)-(A3(1,2)*x(2))-(A3(1,3)*x(3)))/A3(1,1);
% x2=(b3(2)-(A3(2,1)*x1)-(A3(2,3)*x(3)))/A3(2,2);
% x3=(b3(3)-(A3(3,1)*x1)-(A3(3,2)*x2))/A3(3,3);
% x=[x1;x2;x3];
% x1=(b3(1)-(A3(1,2)*x(2))-(A3(1,3)*x(3)))/A3(1,1);
% x2=(b3(2)-(A3(2,1)*x1)-(A3(2,3)*x(3)))/A3(2,2);
% x3=(b3(3)-(A3(3,1)*x1)-(A3(3,2)*x2))/A3(3,3);
% x=[x1;x2;x3];
% x1=(b3(1)-(A3(1,2)*x(2))-(A3(1,3)*x(3)))/A3(1,1);
% x2=(b3(2)-(A3(2,1)*x1)-(A3(2,3)*x(3)))/A3(2,2);
% x3=(b3(3)-(A3(3,1)*x1)-(A3(3,2)*x2))/A3(3,3);
% x=[x1;x2;x3]
```

```
%Jacobi Method
% x=[1;1;1];
% x=[0;0;0]
% x=[2;2;2]
% x=[3;3;3]
% x=[4;4;4]
% x1=(b3(1)-(A3(1,2)*x(2))-(A3(1,3)*x(3)))/A3(1,1);
% x2=(b3(2)-(A3(2,1)*x(1))-(A3(2,3)*x(3)))/A3(2,2);
% x3=(b3(3)-(A3(3,1)*x(1))-(A3(3,2)*x(2)))/A3(3,3);
% x=[x1;x2;x3];
% x1=(b3(1)-(A3(1,2)*x(2))-(A3(1,3)*x(3)))/A3(1,1);
% x2=(b3(2)-(A3(2,1)*x(1))-(A3(2,3)*x(3)))/A3(2,2);
% x3=(b3(3)-(A3(3,1)*x(1))-(A3(3,2)*x(2)))/A3(3,3);
% x=[x1;x2;x3];
% x1=(b3(1)-(A3(1,2)*x(2))-(A3(1,3)*x(3)))/A3(1,1);
% x2=(b3(2)-(A3(2,1)*x(1))-(A3(2,3)*x(3)))/A3(2,2);
% x3=(b3(3)-(A3(3,1)*x(1))-(A3(3,2)*x(2)))/A3(3,3);
% x=[x1;x2;x3];
% x1=(b3(1)-(A3(1,2)*x(2))-(A3(1,3)*x(3)))/A3(1,1);
% x2=(b3(2)-(A3(2,1)*x(1))-(A3(2,3)*x(3)))/A3(2,2);
% x3=(b3(3)-(A3(3,1)*x(1))-(A3(3,2)*x(2)))/A3(3,3);
% x=[x1;x2;x3];
% x1=(b3(1)-(A3(1,2)*x(2))-(A3(1,3)*x(3)))/A3(1,1);
% x2=(b3(2)-(A3(2,1)*x(1))-(A3(2,3)*x(3)))/A3(2,2);
% x3=(b3(3)-(A3(3,1)*x(1))-(A3(3,2)*x(2)))/A3(3,3);
% x=[x1;x2;x3];
% x1=(b3(1)-(A3(1,2)*x(2))-(A3(1,3)*x(3)))/A3(1,1);
% x2=(b3(2)-(A3(2,1)*x(1))-(A3(2,3)*x(3)))/A3(2,2);
% x3=(b3(3)-(A3(3,1)*x(1))-(A3(3,2)*x(2)))/A3(3,3);
% x=[x1;x2;x3];
% x1=(b3(1)-(A3(1,2)*x(2))-(A3(1,3)*x(3)))/A3(1,1);
% x2=(b3(2)-(A3(2,1)*x(1))-(A3(2,3)*x(3)))/A3(2,2);
% x3=(b3(3)-(A3(3,1)*x(1))-(A3(3,2)*x(2)))/A3(3,3);
% x=[x1;x2;x3];
% x1=(b3(1)-(A3(1,2)*x(2))-(A3(1,3)*x(3)))/A3(1,1);
% x2=(b3(2)-(A3(2,1)*x(1))-(A3(2,3)*x(3)))/A3(2,2);
% x3=(b3(3)-(A3(3,1)*x(1))-(A3(3,2)*x(2)))/A3(3,3);
% x=[x1;x2;x3];
% x1=(b3(1)-(A3(1,2)*x(2))-(A3(1,3)*x(3)))/A3(1,1);
% x2=(b3(2)-(A3(2,1)*x(1))-(A3(2,3)*x(3)))/A3(2,2);
% x3=(b3(3)-(A3(3,1)*x(1))-(A3(3,2)*x(2)))/A3(3,3);
% x=[x1;x2;x3];
% x1=(b3(1)-(A3(1,2)*x(2))-(A3(1,3)*x(3)))/A3(1,1);
% x2=(b3(2)-(A3(2,1)*x(1))-(A3(2,3)*x(3)))/A3(2,2);
% x3=(b3(3)-(A3(3,1)*x(1))-(A3(3,2)*x(2)))/A3(3,3);
% x=[x1;x2;x3];
% x1=(b3(1)-(A3(1,2)*x(2))-(A3(1,3)*x(3)))/A3(1,1);
% x2=(b3(2)-(A3(2,1)*x(1))-(A3(2,3)*x(3)))/A3(2,2);
% x3=(b3(3)-(A3(3,1)*x(1))-(A3(3,2)*x(2)))/A3(3,3);
% x=[x1;x2;x3];
% x1=(b3(1)-(A3(1,2)*x(2))-(A3(1,3)*x(3)))/A3(1,1);
% x2=(b3(2)-(A3(2,1)*x(1))-(A3(2,3)*x(3)))/A3(2,2);
% x3=(b3(3)-(A3(3,1)*x(1))-(A3(3,2)*x(2)))/A3(3,3);
% x=[x1;x2;x3]

x = -10:10;
y = -10:10;
```

```
for i = 1:length(x);
    for j = 1:length(y);
        r = [(2*x(i)+3*y(j)-8);(x(i)+4*y(j)-9)];
        res(i,j) = r'*r;
    end
end
contour(x,y,res)
title('Contour Plot for Equation 1')

figure
surf(x,y,res)
title('Surface Plot for Equation 1')

for i = 1:length(x);
    for j = 1:length(y);
        r1 = [(2*x(i)-y(j)-3);(-x(i)+3*y(j)-11)];
        res2(i,j) = r1'*r1;
    end
end
figure
contour(x,y,res2,50)
title('Contour Plot for Equation 2')

figure
surf(x,y,res2)
title('Surface Plot for Equation 2')


G1=A1\b1;
G2=A2\b2;
G3=A3\b3;
A1=[2 3; 1 4]; b1=[8;9];
x0 = [10;10];
pT = [-12;-5];
alpha = 0:.01:1;

for i=1:length(alpha)
        xUpdate = pT*alpha(i)+x0;
        r=b1-A1*xUpdate;
        p(i)=r'*r;
    end
figure;
plot(alpha,p);
title('Linesearch');
xlabel('alpha');
ylabel('phi');

minA= ((A1*(x0+alpha*p)-b1)')*(A1*(x0+alpha*p)-b1);
```