TO: Professor Andreas Linninger / Grant Hartung

FROM: Kehinde Abioye

DATE: September 26, 2017

SUBJECT: Iterative methods for solving linear algebraic equations

The purpose of this assignment is to use the Gauss-Seidel method to solve for systems of linear equations and check those solutions using residual plots.

Introduction

Gauss elimination was used to find the solution to the algebraic system of equations manually. Once the solutions were found by hand, iterative methods were used in MATLAB to verify the solutions. The iterative methods used was Gradient, Gauss-Seidel, and Jacobi methods. A contour map and residual surface map was plotted for the first two set of equations. Five initial guesses were used to show convergence to the same solution, x.

Methods

Part 1:

Gauss elimination was used to solve problems 1-3 by hand

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}; b = \begin{bmatrix} 2 \\ 2 \end{bmatrix} \qquad (1)$$

$$A = \begin{bmatrix} 2 & 1 \\ 1 & 8 \end{bmatrix}; b = \begin{bmatrix} 6 \\ 6 \end{bmatrix} \qquad (2)$$

$$A = \begin{bmatrix} 3 & 1 & 1 \\ 1 & 3 & 1 \\ 0.5 & 1 & 3 \end{bmatrix}; b = \begin{bmatrix} 7 \\ 8 \\ 9.5 \end{bmatrix} \qquad (3)$$

The matrices are combined as such: [A b]. Then, the matrices are reduced to row echelon form by row operations.

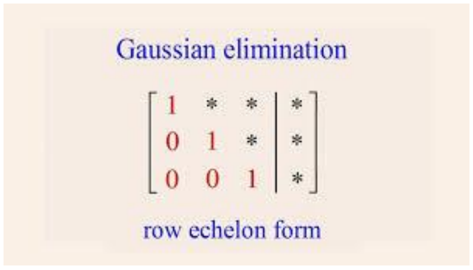
Gaussian elimination
$\left[\begin{array}{ccc|c} 1 & * & * & * \\ 0 & 1 & * & * \\ 0 & 0 & 1 & * \end{array}\right]$
row echelon form

Figure 1 Row echelon form https://www.google.com/search?q=gauss+elimination&rlz=1C1CHBF_enUS786US786&source=lnms&tbm=isch&sa=X&ved=0ahUKEwiozpKUhuffAhUK0KwKHSmZAPEQ_AUIDygC&biw=1242&bih=597#imgrc=ap9vgEN-n6LYwM:

Row operations include row-switching, row multiplication, or row addition.

Part 2:

The contour map of the residual surface is plotted for equations one and two. This was done by setting the x and y values to a range of -10 to 50. Then for loops were created with the equations in residual form. Once in residual form, equation (4) is used to find the contour map of the residual error surface:

$$\varphi(x) = r^T r \qquad (4)$$

Part 3:

The surf function is used to find the residual surface in 3D, using the parameters as explained in part 2.

Part 4:

The Gradient method was used to calculate and plot a contour map for equations one and two. An initial guess of [10;10] was used for x. The code for the gradient method was obtained from the TA.

Results

Problem 1:

The solutions for the equations 1-3 using Gauss elimination is as follows:

x = [2; 1] for equation 1

x = [14/5; 2/5] for equation 2

x = [1; 1.5; 2.5] for equation 3

Problem 2:

A contour plot of residual error for equation 1.

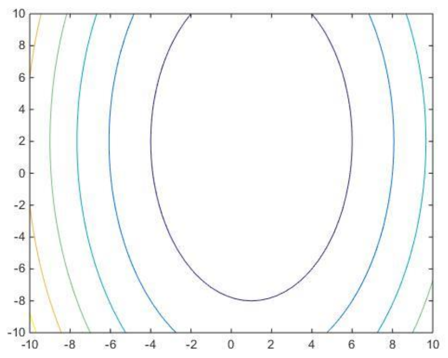

Figure 2 Contour plot for equation 1

A contour plot of residual error for equation 2.

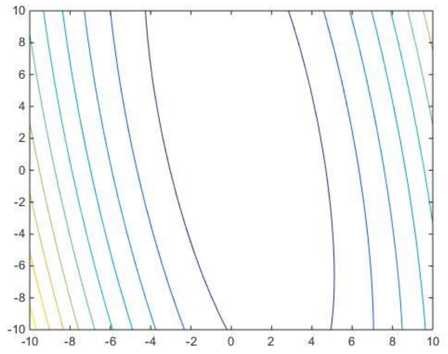

Figure 3 Contour plot for equation 2

Problem 3:

The residual error surface for equation 1.

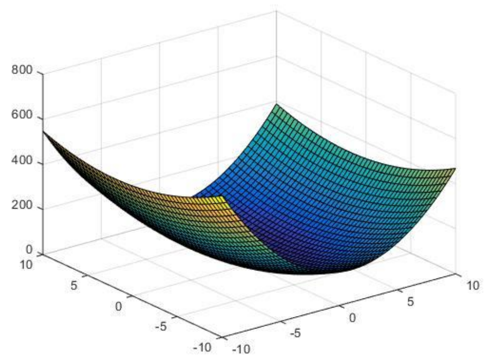

Figure 4 Residual surface for equation 1

The residual error surface for equation 2.

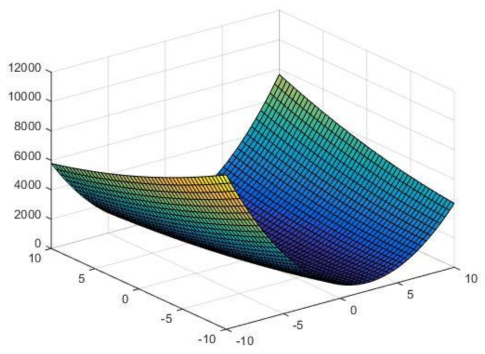

Figure 5 Residual surface for equation 2

Conclusion

Gauss elimination was used to find solutions for the first three problems. Then the Gradient method was used to find the solutions to equations one and two, Gauss-Seidel method was used to find the solutions for equations 1, 2, and 3, and the Jacobi method was used to solve for equation 3. The solutions for each method were the same for each equation. The Gauss-Seidel method required less iterations over the Jacobi method.

Discussion

Gauss elimination can be used to find solutions to less complex equations. The methods used in the homework will come in handy for more complex equations or solutions. Each method has their own benefit.

Appendix A

```
clc

clear all
%
% % Problem 1 and 4
% 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=[1 0; 0 2]; b1=[2;2];
% A2=[2 1;1 8]; b2=[6;6];
% A3=[3 4 2;2 3 1;1 2 1]; b3=[21;14;9];
% Ab1=[A1 b1];
% Ab2=[A2 b2];
% Ab3=[A3 b3];
% x=[10;10];
% x=[0;0;0]
% x=[2;2;2]
% x=[3;3;3]
% x=[1;1;1]
%
% elim1=A1\b1;
% elim2=A2\b2;
% elim3=A3\b3;
%
% n=2;
% tol=1e-6;
%
% 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 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]
%
% %Gauss-Siedel Method for 3
% 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)*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]
%
%
% Gradient Method for 1
% x=[10;10];
% x=[0;0;0]
% x=[2;2;2]
% x=[3;3;3]
% x=[1;1;1]
% GradientMethod(x,b1,A1)
% GradientMethod(x,b2,A2)
%
% 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]
%
% Problem 2 and 3
% Plot
% xp=linspace(-10,10,50);
% y=xp;
%
% for i=1:length(xp)
%     for j=1:length(y)
%         r=(A1*[xp(i);y(j)])-b1;
%         p(i,j)=r'*r;
%     end
% end
%
%
% figure;
% surf(xp,y,p)
% title('Surface Plot for Equation 1');
% xlabel('X');
% ylabel('Y');
% zlabel('Z');
%
% figure;
% contour(xp,y,p)
% title('Contour Plot for Equation 1');
% xlabel('X');
% ylabel('Y');
% zlabel('Z');
%
% for i=1:length(xp)
%     for j=1:length(y)
%         r=(A2*[xp(i);y(j)])-b2;
%         p(i,j)=r'*r;
%     end
% end
%
%
% figure;
% surf(xp,y,p)
% title('Surface Plot for Equation 2');
% xlabel('X');
% ylabel('Y');
% zlabel('Z');
%
% figure;
% contour(xp,y,p)
% title('Contor Plot for Equation 2');
% xlabel('X');
% ylabel('Y');
% zlabel('Z');
```