TO: Professor Andreas Linninger / Grant Hartung

FROM: Kehinde Abioye

DATE: 10/26/2017

SUBJECT: Consolidation of Iterative Methods

To practice different types of iterative methods to better understand how iterative solving works.

Introduction

This assignment focuses on solving equations with iterative methods as done in other assignments. Two new iterative methods are introduced: Conjugate Gradient and Newton-Raphson methods. The Gauss-Seidal method was also used. Convergence histories were also plotted with different initial x.

Methods

Part 1

Part a:

Equations 1 and 2 were solved by hand using Gauss-Seidal method.

$$x_1 + 3x_2 = -3 \qquad (1)$$
$$2x_1 + x_2 = -1 \qquad (2)$$

The matrices must be in diagonal dominance to solve it so equation 1 and 2 were switched ([eq2;eq1]). The initial guess is x =[1;1]. To solve, equation 2 was solved for x1 and equation 1 was solved for x2. The initial x was inputted in the equations and x1 and x2 was updated for each time it was solved. Table 1 was updated after each iteration up to 4 iterations.

k	$\mathbf{x_1^{(k)}}$	$\mathbf{x_2^{(k)}}$
0		
1		
2		
3		
4		

Table 1. Solution of $\boldsymbol{x}$ for each iteration, k.

Part b:

X1 and X2 were plotted and the x updates were plotted as well on the same plot using the plot command in MATLAB.

Part c:

The solution was verified using "A\b" in MATLAB.

Part 2

Part a:

Equations 3 and 4 were solved in MATLAB using Gauss-Seidel Method.

$$\begin{bmatrix} 4 & 1 & -1 \\ 2 & 7 & 1 \\ 1 & -3 & 12 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 7 \\ 15 \\ -17 \end{bmatrix} \quad (3)$$

$$\begin{bmatrix} 10 & -1 & 2 & 0 \\ -1 & 11 & -1 & 3 \\ 2 & -1 & 10 & -1 \\ 0 & 3 & -1 & 8 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 6 \\ 25 \\ -11 \\ 15 \end{bmatrix} \quad (4)$$

A Gauss-Seidel function was made which consisted of for loops which helped iterate the values, as well as calculate the residual error. The initial guesses were 1,2,3,4,and 5.

Part b:

The plot of the evolution of the total residual error was made for each initial guess for both equations. The plot was residual error vs. iterations.

Part c:

The solution was verified using "A\b".

Part 3:

Part a:

Equations 1 and 2 were solved in MATLAB using Gradient Method. A contour plot of the energy surface was created with a for loop iterating the residual error and plotting it against an x range of -20 to 20. An initial guess of [-8;8] was used.

Part b:

Four iterations were performed and table 2 was updated for each iteration:

k	$\mathbf{x^{(k)}}$	$\boldsymbol{\alpha^{(k)}}$	$\mathbf{r(x^k)}$	$\mathbf{r^T(x^k)r(x^k)}$
0				
1				
2				
3				
4				

Table 2. Solution of Gradient/Conjugate Gradient Method for each iteration, k

Part c:

The x updates were plotted on the contour plot for Gradient Method.

Part 4:

Part a:

Equations 3 and 4 were solved in MATLAB using Gradient Method with initial guesses x = [10;-10;10] and x= [0;0;0;0] respectively. Four iterations were calculated using the Gradient function.

Part b:

Table 2 was used to record the values requested.

Part c:

The solution was verified using A\b in MATLAB.

Part 5

Part b:

The Conjugate Gradient Method was used for equation 2 and the values listed in a table like table 2.

Part 6

Equation 4 was solved using Newton-Raphson method with initial guess x = [1;1]. A residual error surface was plotted for x = [-1;-1] to [2;2] with [.01;.01] increments using for loops.

Results

Part 1

Solve by hand using Gauss-Seidel Method for system of equations (1):

$$\begin{aligned} x_1 + 3x_2 &= -3 \\ 2x_1 + x_2 &= -1 \end{aligned} \quad (1)$$

k	$x_1^{(k)}$	$x_2^{(k)}$
0	1	1
1	-1	-0.667
2	-0.167	-0.944
3	-0.0278	-0.991
4	-0.00463	-0.998

Table 3. Solution of x for equation 1 for iteration, k.

Figure 2 shows the history of vector x on the graph of x1 and x2.

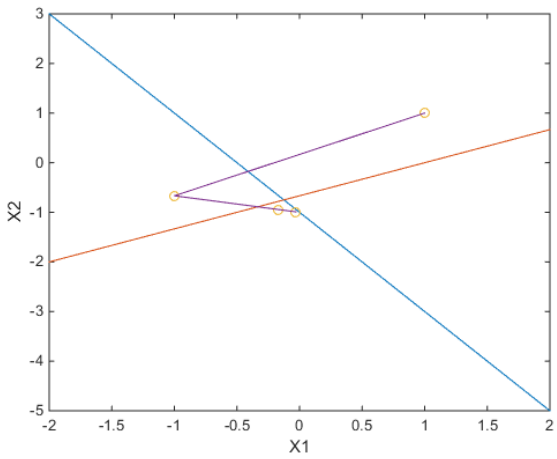


Figure 1 History of updates in x for the first set of equations

Part 2

Solve in MATLAB using Gauss-Seidel Method for system of equations (2) and (3):

$$\begin{bmatrix} 4 & 1 & -1 \\ 2 & 7 & 1 \\ 1 & -3 & 12 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 7 \\ 15 \\ -17 \end{bmatrix} \quad (2)$$

$$\begin{bmatrix} 10 & -1 & 2 & 0 \\ -1 & 11 & -1 & 3 \\ 2 & -1 & 10 & -1 \\ 0 & 3 & -1 & 8 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 6 \\ 25 \\ -11 \\ 15 \end{bmatrix} \quad (3)$$

Evolution of total residual error for (2):

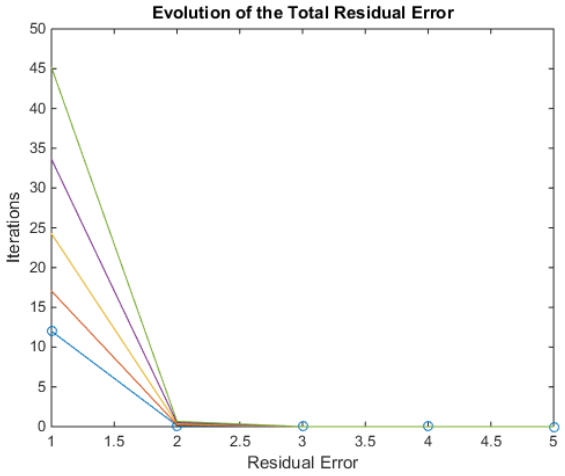
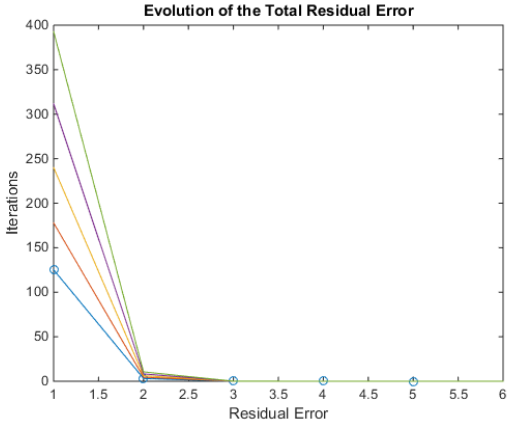


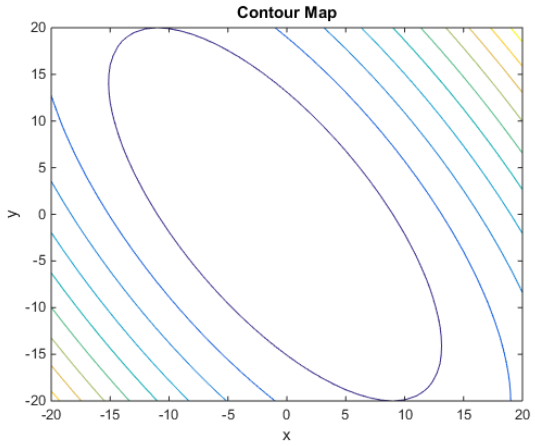
Figure 2 Evolution of total residual error for set of equations (2)

Evolution of total residual error for (3):



Part 3

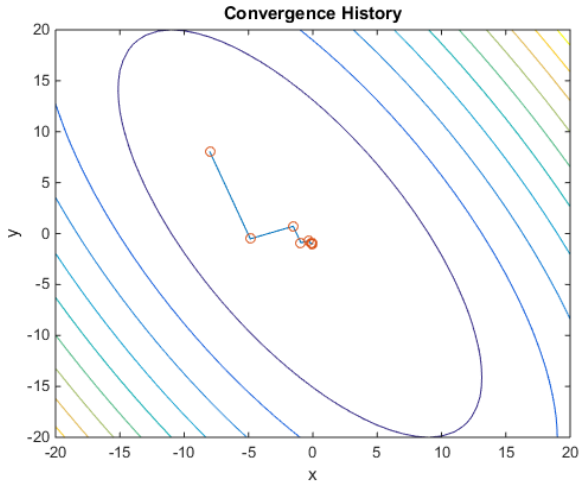
Contour map of the energy surface for equation (1) and (2):



k	$\mathbf{x^{(k)}}$	$\boldsymbol{\alpha^{(k)}}$	$\mathbf{r(x^k)}$	$\mathbf{r^T(x^k)r(x^k)}$
0	[-8; 8]	0.448	[9.2; -3.4]	96.97
1	[-4.86; -0.51]	0.361	[1.3; -3.6]	14.91
2	[-1.53; 0.72]	0.448	[1.76; 0.649]	3.53
3	[-0.93; -0.91]	0.361	[0.25; -0.69]	0.542
4	[-0.29; -0.67]	0.448	[0.336; 0.124]	0.128

Table 4. Solution of Gradient/Conjugate Gradient Method for each iteration, *k*, for equation 1

Convergence history on contour map:



Part 4:

k	$\mathbf{x^{(k)}}$	$\boldsymbol{\alpha^{(k)}}$	$\mathbf{r(x^k)}$	$\mathbf{r^T(x^k)r(x^k)}$
0	[10; -10; 10]	0.0832	[-27.9; 39.9; 14.4]	2.6e3
1	[8.92; -5.43; -4.72]	0.210	[-9.8; -10.1; 9.02]	279.6
2	[3.03; 2.96; -1.68]	0.109	[-3.4; -1.2; -5.1]	38.8
3	[1.96; 1.85; -0.69]	0.106	[-2.4; 0.95; 1.4]	8.36
4	[1.6; 1.7; -1.2]	0.201	[-0.38; 0.29; -0.8]	0.99

Table 5. Solution of Gradient Method for each iteration, *k,* for equation 2

k	$\mathbf{x^{(k)}}$	$\boldsymbol{\alpha^{(k)}}$	$\mathbf{r(x^k)}$	$\mathbf{r^T(x^k)r(x^k)}$
0	[0; 0; 0; 0]	0.0786	[4.9; -0.55; -0.15; -1.2]	2.6e3
1	[0.47; 1.96; -0.86; 1.1]	0.0988	[0.04; 0.88; -1.2; -0.1]	279.6
2	[0.96; 1.9; -0.88; 1.1]	0.0923	[0.29; -0.09; -0.02; -0.38]	38.8
3	[0.96; 1.99; -0.99; 1.1]	0.103	[-0.01; 0.16; -0.11; -0.04]	8.36
4	[0.99; 1.99; -0.99; 1.01]	0.095	[0.04; -0.008; 0.009; -0.06]	0.99

Table 6. Solution of Gradient Method for each iteration, *k,* for equation 3

Part 5:

Part 5a:

The Conjugate Gradient Method is when there is a symmetric positive definite matrix.

Part 5b:

k	$\mathbf{x^{(k)}}$	$\boldsymbol{\alpha^{(k)}}$	$\mathbf{r(x^k)}$	$\mathbf{r^T(x^k)r(x^k)}$
0	[10; -10; 10]	0.0832	[-27.9; 39.9; 14.4]	2.58e3
1	[8.9; -5.4; -4.7]	0.199	[-13.5; -10.1; 43.5]	2.18e3
2	[3.17; 3.32; -4.48]	0.0828	[-0.89; -18.3; 6.3]	374.4
3	[0.59; 4.7; -0.82]	0.129	[1.89; -1.76; -7.89]	69.03
4	[0.65; 2.24; -0.252]	0.0943	[1.07; 2.63; -2.07]	12.36

Table 7. Solution of Conjugate Gradient Method for each iteration, *k,* for equation 2

Part 6

Residual error surface for equation 4:

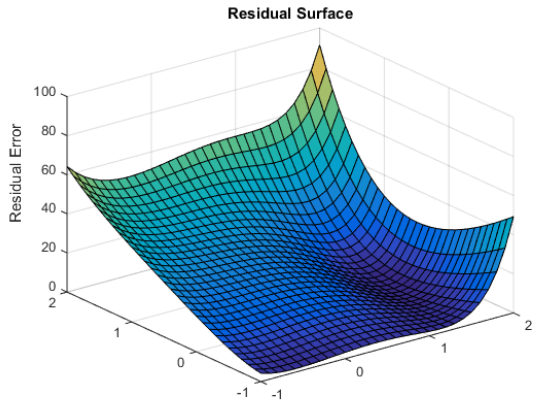


Figure 3 Residual error surface for equation 4

Conclusion

Iterative methods were used to find solutions for system of equations. Newton-Raphson was a new iterative method introduced. Gauss-Seidel and Gradient were also used. Gauss-Seidal was also done by hand.

Discussion

Iterative methods have proven its importance once again. Solving system of equations is important in modeling various biological systems.

Appendix A

```
clc
clear all
close all

%%Part 1
A=[2 1;1 3]; b=[-1;-3];
C=[A b];

x=linspace(-2,2,9);
y=-1-(2*x);
y2=(-3-y)/3;

xhistory=[1;-1;(-1/6);(-1/36)];
yhistory=[1;(-2/3);(-17/18);(-107/108)];

plot(x,y)
hold on
plot(x,y2)
hold on
plot(xhistory,yhistory,'o')
hold on
plot(xhistory,yhistory)
xlabel('X1');ylabel('X2')

verify=A\b;

%%Part 2
A2=[4 1 -1;2 7 1;1 -3 12]; b2=[7;15;-17];
A3=[-10 -1 2 0;-1 11 -1 3;-2 -1 10 -1;0 3 -1 8]; b3=[6;25;-11;-15];
C2=[A2 b2];
C3=[A3 b3];
x0=[1;1;1];
x10=[2;2;2];
x20=[3;3;3];
x30=[4;4;4];
x40=[5;5;5];
nmax=1000;
tol=1e-6;
omega=1;


% function [x,iter]=Gauss(A,b,x0,nmax,tol,omega)
% [n,m]=size(A);
% if n ~= m, error('Only squared systems'); end
% iter=0; r=b-A*x0; r0=norm(r); err=norm(r); xold=x0;
% while err > tol & iter < nmax
% iter = iter + 1;
% for i=1:n
% s=0;
% for j = 1:i-1, s=s+A(i,j)*x(j); end
% for j = i+1:n, s=s+A(i,j)*xold(j); end
% x(i,1)=omega*(b(i)-s)/A(i,i)+(1-omega)*xold(i);
% end
```

```
% xold=x; r=b-A*x; err=norm(r)/r0;
% end
% return


%Part A equations 2
[xHist,xfinal,iters,r,err]= Gauss(A2,b2,x0,nmax,tol,omega)
[xHist,xfinal2,iters2,r2,err2]= Gauss(A2,b2,x10,nmax,tol,omega)
[xHist,xfinal3,iters3,r3,err3]= Gauss(A2,b2,x20,nmax,tol,omega)
[xHist,xfinal4,iters4,r4,err4]= Gauss(A2,b2,x30,nmax,tol,omega)
[xHist,xfinal5,iters5,r5,err5]= Gauss(A2,b2,x40,nmax,tol,omega)

%Part B equations 2
plot(1:iters,err,'-o')
hold on
plot(1:iters2,err2)
hold on
plot(1:iters3,err3)
hold on
plot(1:iters4,err4)
hold on
plot(1:iters5,err5)
xlabel('Residual Error')
ylabel('Iterations')
title('Evolution of the Total Residual Error')

%Part A equations 3
x0=[1;1;1;1];
x10=[2;2;2;2];
x20=[3;3;3;3];
x30=[4;4;4;4];
x40=[5;5;5;5];

[xHist,xfinal,iters,r,err]= Gauss(A3,b3,x0,nmax,tol,omega)
[xHist,xfinal2,iters2,r2,err2]= Gauss(A3,b3,x10,nmax,tol,omega)
[xHist,xfinal3,iters3,r3,err3]= Gauss(A3,b3,x20,nmax,tol,omega)
[xHist,xfinal4,iters4,r4,err4]= Gauss(A3,b3,x30,nmax,tol,omega)
[xHist,xfinal5,iters5,r5,err5]= Gauss(A3,b3,x40,nmax,tol,omega)

%Part B eqution 3
plot(1:iters,err,'-o')
hold on
plot(1:iters2,err2)
hold on
plot(1:iters3,err3)
hold on
plot(1:iters4,err4)
hold on
plot(1:iters5,err5)
xlabel('Residual Error')
ylabel('Iterations')
title('Evolution of the Total Residual Error')

%Part C
verify2=A2\b2;
```

```
verify3=A3\b3;

%%Part 3
x0 = [-8;8];
A = [2 1;1 3];
b = [-1;-3];
tol=1e-6;

% function [xHistory,rNew,alphaR,resR,surfRes]=gradient1(A,b,x,tol)
% xHistory(:,1) = x; %Store initial guess
% r = b-A*x; %Compute first residual
% for i = 2:100
%  alpha = r'*r/(r'*A*r);
%  xHistory(:,i) = x + alpha*r;
%  rNew = r - alpha*A*r;
%  if norm(rNew) < tol % using the norm
%  if norm(rNew,inf) < tol % using the maxNorm
%  xHistory(:,i), break
%  end
%  end
%  x = xHistory(:,i); r = rNew; %update variables
%  alphaR(:,i-1)=alpha;
%  resR(:,i-1) = r;
% surfRes(:,i-1) = r'*r;
% end
% return

[xHistory,rNew,alphaR,resR,SurfRes]=gradient1(A,b,x0,tol)

xVal = -20:20;
yVal = -20:20;

for i = 1:length(xVal);
    for j = 1:length(yVal);
        res = [xVal(i)+(3*yVal(j))+3;(2*xVal(i))+yVal(j)+1];
        resM(i,j)= res'*(res);
    end
end

figure;
contour(xVal,yVal,resM)
title('Contour Map');
xlabel('x');
ylabel('y');

figure;
contour(xVal,yVal,resM)
hold on
plot(xHistory(1,:),xHistory(2,:));
hold on
plot(xHistory(1,:),xHistory(2,:),'o')
title('Convergence History');
xlabel('x');
ylabel('y');
```

```
%Part 4
clc
clear all
A2=[4 1 -1;2 7 1;1 -3 12]; b2=[7;15;-17];
x02 = [10;-10;10];
A3=[-10 -1 2 0;-1 11 -1 3;-2 -1 10 -1;0 3 -1 8]; b3=[6;25;-11;-15];
x03 = [0;0;0;0];
tol=1e-6;

[xHistory,rNew,alphaR,resR,SurfRes]=gradient1(A2,b2,x02,tol)
[xHistory,rNew,alphaR,resR,SurfRes]=gradient1(A3,b3,x03,tol)

%Part 5
A2=[4 1 -1;2 7 1;1 -3 12]; b2=[7;15;-17];
x02 = [10;-10;10];
tol = 1e-6;

% function [xNew,alphaNew,resid,ResidSurf,itr,newp]=conjgrad(A,b,x,tol)
% xNew(:,1) = x;
% r = b-A*x; p=r; itr =0;
% for i=2:100
% itr = itr+1;
% alpha = (p'*r)/(p'*A*p);
% xNew(:,i) = x+(alpha*p);
% rNew = r-(alpha*A*p);
% beta = ((A*p)'*rNew)/((A*p)'*p);
% Pnew = rNew-(beta*p);
% if norm(rNew)<tol
% if norm(rNew,inf)<tol;
% xNew(:,i),itr, break
% end
% end
% x = xNew(:,i); r = rNew; alphaNew(:,i) = alpha;
% resid(:,i) = r; newp(:,i)= p; p = Pnew; ResidSurf(:,i) = r'*r;
% end

[xNew2,alphaNew,resid,residSurf,itr,newp]=conjgrad(A2,b2,x02,tol)

%Part 6
x = -1:.1:2;
y= -1:.1:2;

for i=1:length(x);
    for j=1:length(y);
        resN = [x(i)^2+x(i)*y(j)+y(j)^2-2.5;2*x(i)-y(j)^3+3];
        resSurf(i,j) = resN'*resN;
    end
end

figure;
surf(x,y,resSurf)
title('Residual Surface');
zlabel('Residual Error');
```