Ryan Brosnahan
Homework 3

1. Implement the back-substitution algorithm (backsub) in Section 3.3 and apply it to the system

$$\begin{bmatrix} 5 & -2 & 1 & 15 \\ 0 & -5 & 1607 & -3 \\ 0 & 0 & 4 & 4 \\ 0 & 0 & 0 & 21 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$$

Turn in a listing of the code, as well as the output.

```
%backsub.m

function x = backsub(U, Y)
%Y=UX U:NxN Y:Nx1

N = length(Y);
x = zeros(N,1);
x(N)  = Y(N)/U(N,N);

for k=N-1:-1:1
    x(k)  = (Y(k)-U(k,k+1:N)*x(k+1:N))/U(k,k);
end
```

```
>> U=[5 -2 1 15; 0 -5 1607 -3; 0 0 4 4; 0 0 0 21]
>> Y = [1; 2; 3; 4]
>> backsub(U, Y)

ans =

   7.124333333333334e+01
   1.793166666666667e+02
   5.595238095238095e-01
   1.904761904761905e-01
```

2. Implement the algorithm on page 136-7, called uptrbk.m, which employs upper triangularization with partial pivoting. Apply it to the system

$$\begin{bmatrix} 5 & -2 & 1 & 15 \\ -3 & -5 & 1607 & -3 \\ 10 & 2 & 4 & 4 \\ 4 & 1.2 & 9.8 & 21 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 7 \\ -2 \\ 33 \\ 4 \end{bmatrix}$$

From your website:

```
%uptrbk.m

function X = uptrbk(A,B)
%Ax=B; N is NxN; B is Nx1

[N,N] = size(A);
X = zeros(N,1);
C=zeros(1,N+1);
Aug = [A B];
for p=1:N-1
   [Y,j] = max(abs(Aug(p:N,p)));
   C = Aug(p,:);
   Aug(p,:) = Aug(j+p-1,:);
   Aug(j+p-1,:) = C;
   if Aug(p,p)==0
      'A was singular.  No unique solution'
      break
   end

   for k=p+1:N
      m = Aug(k,p)/Aug(p,p);
      Aug(k,p:N+1) = Aug(k,p:N+1)-m*Aug(p,p:N+1);
   end
end

X = backsub(Aug(1:N,1:N),Aug(1:N,N+1));
```

```
>> A = [5 -2 1 15; -3 -5 1607 -3; 10 2 4 4; 4 1.2 9.8 21]
>> B = [7; -2; 33; 4]
>> uptrbk(A, B)

ans =

    3.295073687199341e+00
    1.006377669420495e+00
    7.108379037624037e-03
   -4.979814317462225e-01
```

3. Let's play with the idea of ill-conditioning for a 9×9 system $Ax = B$. First, create A that is all ones except for 1.01's on the main diagonal:

```
A = ones(9,9);
for j=1:9
  A(j,j) = 1.01;
end
```

Let $B = (1:9)'$.

(a) Solve the system using the backslash command and call the result $x1$.

```
>> x1 = A\B

x1 =

  -3.994450610432928e+02
  -2.994450610432837e+02
  -1.994450610432854e+02
  -9.944506104328704e+01
   5.549389567171661e-01
   1.005549389567171e+02
   2.005549389567170e+02
   3.005549389567170e+02
   4.005549389567133e+02
```

(b) Perturb the matrix A slightly by multiplying $A(5,3)$ by 1.02 (a 2% change):

```
A(5,3) = A(5,3)*1.02
```

Solve the system using the backslash command and call the result $x2$.

```
>> x2 = A\B

x2 =

  -4.563480741797438e+02
  -3.563480741797438e+02
  -2.563480741797447e+02
  -1.563480741797431e+02
   4.563480741797473e+02
   4.365192582025690e+01
   1.436519258202578e+02
   2.436519258202556e+02
   3.436519258202554e+02
```

(c) Compare $x1$ and $x2$ by printing each value and the absolute value of the difference for each $j = 1 : 9$. Then check the overall effect by computing the relative error:

```
norm(x1-x2)/norm(x1)
```

Answer the question: A 2% change in one element causes a —% change in the overall solution. Is there evidence of ill-conditioning? If there is, how would you rate the severity of the problem?

```
for j=1:9
disp('x1 at ')
j
disp(' ')
x1(j)
disp('x2 at ')
j
x2(j)
disp('abs err: ')
abs(x1(j)-x2(j))
end
```

Condensed to a table:

x1	x2	abs(x1 - x2)	abs(x1-x2)/abs(x1)
-3.99445061043E+02	-4.5634807418E+02	5.69030131365E+01	1.42455167646E-01
-2.99445061043E+02	-3.5634807418E+02	5.69030131365E+01	1.90028224003E-01
-1.99445061043E+02	-2.5634807418E+02	5.69030131365E+01	2.85306704708E-01
-9.94450610433E+01	-1.5634807418E+02	5.69030131365E+01	5.72205522723E-01
5.54938956717E-01	4.5634807418E+02	4.55793135223E+02	8.21339229668E+02
1.00554938957E+02	4.3651925820E+01	5.69030131365E+01	5.65889788476E-01
2.00554938957E+02	1.4365192582E+02	5.69030131365E+01	2.83727807615E-01
3.00554938957E+02	2.4365192582E+02	5.69030131365E+01	1.89326494963E-01
4.00554938957E+02	3.4365192582E+02	5.69030131365E+01	1.42060445652E-01

There is evidence of extreme ill-conditioning. Just a 2% change in one parameter propagated into error of massive proportions for all parameters (>14%).

```
>> norm(x1-x2)/norm(x1)
ans =
   6.240324858473382e-01

>> norm(x1-x2)
ans =
   4.833746014722979e+02
```

These are pretty big which suggests ill-conditioning.

(d) Now consider $Cx = B$, where B is the same, but C is a 9×9 matrix that is all ones except for 10's on the main diagonal. Solve the system and call the solution $y1$.

```
>> y1 = C\B
y1 =
  -1.666666666666667e-01
  -5.555555555555555e-02
   5.555555555555550e-02
   1.666666666666666e-01
   2.777777777777778e-01
   3.888888888888890e-01
   5.000000000000000e-01
   6.111111111111114e-01
   7.222222222222221e-01
```

(e) Perturb the matrix by multiplying $C(5,3)$ by 1.02. Solve the system and call the solution $y2$.

```
>> y2 = C\B
y2 =
  -1.666598071092453e-01
  -5.554869599813423e-02
   5.556241511297687e-02
   1.666735262240880e-01
   2.776611653016147e-01
   3.888957484463103e-01
   5.000068595574214e-01
   6.111179706685326e-01
   7.222290817796437e-01
```

(f) Repeat part c. A 2% change in one element causes a —% change in the overall solution. Is there evidence of ill-conditioning? If there is, how would you rate the severity of the problem?

y1	y2	abs(y1 - y2)	abs(y1-y2)/abs(y1)
-1.6666666667E-01	-1.6665980711E-01	6.85955742100E-06	4.11573445260E-05
-5.5555555556E-02	-5.5548695998E-02	6.85955742129E-06	1.23472033583E-04
5.5555555556E-02	5.5562415113E-02	6.85955742130E-06	1.23472033583E-04
1.6666666667E-01	1.6667352622E-01	6.85955742200E-06	4.11573445320E-05
2.7777777778E-01	2.7766116530E-01	1.16612476163E-04	4.19804914187E-04
3.8888888889E-01	3.8889574845E-01	6.85955742097E-06	1.76388619396E-05
5.0000000000E-01	5.0000685956E-01	6.85955742097E-06	1.37191148419E-05
6.1111111111E-01	6.1111797067E-01	6.85955742097E-06	1.12247303252E-05
7.2222222222E-01	7.2222908178E-01	6.85955742097E-06	9.49784873673E-06

There is no evidence of ill conditioning, the 2% change led to a <.014% change in the solutions. That's pretty small!

```
>> norm(y1-y2)/norm(y1)
ans =
   9.867800240768174e-05

>> norm(y1-y2)
ans =
   1.182154720035904e-04
```

These are small. Although it isn't a perfect measure, and ignoring all other evidence, it is likely based on the norm that these are not ill-conditioned.

(g) Now compute the condition numbers of the unperturbed matrices A and C to see how they compare.

```
>> norm(C)
ans =
  18

>> norm(A)
ans =
   9.01
```

These results are counter-intuitive considering C was less affected by perturbation than A.

4. Use the Jacobi iteration algorithm to solve the system

$$\begin{aligned} x - 5y - z &= -8 \\ 4x + y - z &= 13 \\ 2x - y - 6z &= -2 \end{aligned}$$

Remember that you might need to modify the system to put it into the appropriate form. Use 2 different starting guesses, $(0, 0, 0)'$ and $(10, 20, -30)'$, to see how many iterations it takes to converge. Use the error tolerance tol $= 1e-10$. Turn in a listing of the code, as well as the output. Include the number of iterations in your output.

Just to check:

```
>> A\B
ans =
   3.000000000000000e+00
   2.000000000000000e+00
   1.000000000000000e+00
```

From your website:

```
%jacobi.m

function [X,k] = jacobi(A,B,P,delta,max1)
N = length(B);
for k=1:max1
   for j=1:N
      X(j)=(B(j)-A(j,[1:j-1,j+1:N])*P([1:j-1,j+1:N]))/A(j,j);
   end
   err = norm(X'-P);
   relerr = err/(norm(X)+eps);
   P = X';
   fprintf('%4g  %20.15f  %20.15f  %20.15f\n',k,X(1),X(2),X(3))
   if (err<delta)|(relerr<delta)
      break
   end
end
X = X';
```

First guess (0,0,0)’ takes 19 iterations:

```
>> jacobi(A, B, guess, 1e-10, 100)
ans =
   2.999999999942206e+00
   2.000000000005136e+00
   9.999999999343087e-01
```

Second guess (10,20,-30)’ takes 23 iterations:

```
>> jacobi(A, B, guess2, 1e-10, 100)
ans =
   2.999999999960004e+00
   2.000000000003563e+00
   9.999999999545466e-01
```

5. Now use the Gauss-Seidel algorithm on the problem above. Use the same 2 starting guesses. Use the error tolerance tol $= 1e - 10$. Turn in a listing of the code, as well as the output. Include the number of iterations in your output.

```
%gseid.m

function [X,k]=gseid(A,B,P,delta,max1)
N=length(B);
X=zeros(N,1);
for k=1:max1
   for j=1:N
      X(j) = (B(j)-A(j,1:j-1)*X(1:j-1) - A(j,j+1:N)*P(j+1:N))/A(j,j);
   end
   err = abs(norm(X-P));
   relerr = err/(norm(X)+eps);
   P = X;
   fprintf('%4g  %20.15f %20.15f  %20.15f\n',k,X(1),X(2),X(3))
   if (err<delta)|(relerr<delta)
      break
   end
end
X = X';
```

First guess (0,0,0)' takes 13 iterations:

```
>> gseid(A, B, guess, 1e-10, 100)

ans =
   2.999999999969199e+00    2.000000000008718e+00    9.999999999882799e-01
```

Second guess (10,20,-30)' takes 16 iterations:

```
>> gseid(A, B, guess2, 1e-10, 100)

ans =

   2.999999999983042e+00    2.000000000004545e+00    9.999999999935897e-01
```

6.

Consider tic toc functions