Lab 3 notes

First we use matlab to make the cubic spline we covered in class. This is a spline for $\cos(\pi x)$ with interpolation points $x = 0, .25, .5, .75, 1$.

Here is the diary transcript from a session much like that covered in the lab.

```
octave:2> x=linspace(0,1,5)'
x =

  0.00000
  0.25000
  0.50000
  0.75000
  1.00000

octave:3> y=cos(pi.*x)
y =

    1.0000e+00
    7.0711e-01
    6.1230e-17
   -7.0711e-01
   -1.0000e+00

octave:4> A=zeros(5)
A =

  0  0  0  0  0
  0  0  0  0  0
  0  0  0  0  0
  0  0  0  0  0
  0  0  0  0  0

octave:5> A(1,1)=1;
octave:6> A(5,5)=1;
octave:7> A
A =

  1  0  0  0  0
  0  0  0  0  0
  0  0  0  0  0
  0  0  0  0  0
  0  0  0  0  1

octave:8> for i=2:4
> A(i,i-1)=.25;
> A(i,i)=1;
> A(i,i+1)=.25;
> end
octave:9> A
A =

  1.00000  0.00000  0.00000  0.00000  0.00000
  0.25000  1.00000  0.25000  0.00000  0.00000
  0.00000  0.25000  1.00000  0.25000  0.00000
  0.00000  0.00000  0.25000  1.00000  0.25000
  0.00000  0.00000  0.00000  0.00000  1.00000
```

```
octave:10> F=zeros(5,1)
F =

  0
  0
  0
  0
  0

octave:11> for i=2:4
>    F(i)=3*((y(i+1)-y(i))/(x(i+1)-x(i))-(y(i)-y(i-1))/(x(i)-x(i-1)));
> end
octave:12> F
F =

   0.00000
  -4.97056
  -0.00000
   4.97056
   0.00000

octave:13> c=inv(A)*F
c =

   0.00000
  -4.97056
  -0.00000
   4.97056
   0.00000

octave:14> d=zeros(4,1)
d =

  0
  0
  0
  0

octave:15> for i=1:4
> d(i)=(c(i+1)-c(i))/.75;
> end
octave:16> d
d =

  -6.6274
   6.6274
   6.6274
  -6.6274

octave:17> b=zeros(4,1)
b =

  0
  0
  0
  0
```

```
octave:18> for i=1:4
> b(i)=(y(i+1)-y(i))/(x(i+1)-x(i))-c(i)*.25-d(i)*.25^2;
> end
octave:19> b
b =

  -0.75736
  -2.00000
  -3.24264
  -2.00000

octave:20> a=y
a =

    1.0000e+00
    7.0711e-01
    6.1230e-17
   -7.0711e-01
   -1.0000e+00

octave:21> xi=zeros(4,25);
octave:22> for i=1:4
>   xi(i,:)=linspace(x(i),x(i+1),25);
> end
octave:23> s=zeros(4,25);
octave:24> for i=1:4
> s(i,:)=a(i)+b(i).*(xi(i,:)-x(i))+c(i).*(xi(i,:)-x(i)).^2+d(i).*(xi(i,:)-x(i)).^3;
> end
octave:25> plot(xi(1,:),s(1,:))
octave:26> hold on
octave:27> plot(xi(2,:),s(2,:))
octave:28> plot(xi(2,:),s(2,:),'r')
octave:29> plot(xi(3,:),s(3,:),'g')
octave:30> plot(xi(4,:),s(4,:),'b')
octave:31> gset term postscript eps
octave:32>
octave:32> gset output "lab3plot1.eps"
octave:33>
octave:33> gset term postscript eps
octave:34> diary off
```

Your program can work in a similar manner to this code. It should take two vectors as input. It can also take the length of the vectors as input, or it can find the length using the command $n = length(x)$. The program should then output 4 vectors (the a,b,c,d vectors). If you wish you can make a separate program that takes in the a,b,c,d vectors and some points x and makes a plot of the spline. You can also do this by hand. Anyway the first line of you function should look either like.

```
function [a,b,c,d]=spline(x,y,n)
```

or

```
function [a,b,c,d]=spline(x,y)

n=length(x);
```

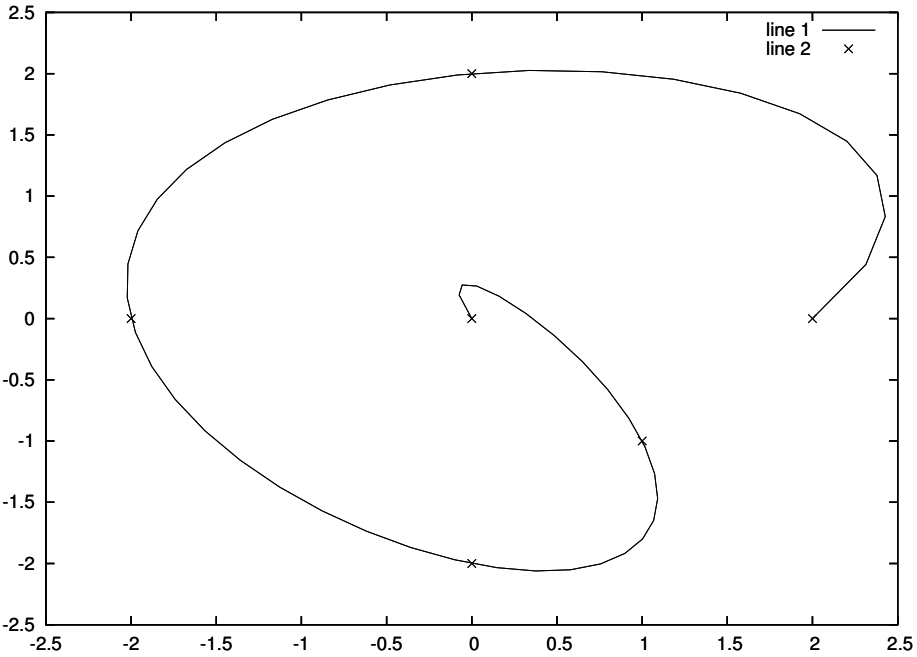
In the example that I did, n was 4.

Now we will look at finding and plotting parametric curves and Bezier splines. I will assume you have written a function called divdif to produce a divided difference table. Here is a copy of the session from the lab (with some of the errors and extra output deleted).

```
octave:2> x=[2;0;-2;0;1;0]
x =

   2
   0
  -2
   0
   1
   0

octave:3> y=[0;2;0;-2;-1;0]
y =

   0
   2
   0
  -2
  -1
   0

octave:4> t=linspace(0,1,length(x))'
t =

  0.00000
  0.20000
  0.40000
  0.60000
  0.80000
  1.00000

octave:5> Fx=divdif(t,x)
Fx =

     2.00000     0.00000     0.00000     0.00000     0.00000     0.00000
     0.00000   -10.00000     0.00000     0.00000     0.00000     0.00000
    -2.00000   -10.00000     0.00000     0.00000     0.00000     0.00000
     0.00000    10.00000    50.00000    83.33333     0.00000     0.00000
     1.00000     5.00000   -12.50000  -104.16667  -234.37500     0.00000
     0.00000    -5.00000   -25.00000   -20.83333   104.16667   338.54167

octave:6> Fy=divdif(t,y)
Fy =

     0.00000     0.00000     0.00000     0.00000     0.00000     0.00000
     2.00000    10.00000     0.00000     0.00000     0.00000     0.00000
     0.00000   -10.00000   -50.00000     0.00000     0.00000     0.00000
    -2.00000   -10.00000     0.00000    83.33333     0.00000     0.00000
    -1.00000     5.00000    37.50000    62.50000   -26.04167     0.00000
     0.00000     5.00000     0.00000   -62.50000  -156.25000  -130.20833

octave:7> [tt,xx]=divdifplot(t,Fx);
octave:8> [tt,yy]=divdifplot(t,Fy);
octave:9> plot(xx,yy)
octave:10> hold on
octave:11> plot(x,y,'+')
octave:12> gset output "lab3-07a.ps"
octave:13> gset term postscript eps
```

```
octave:14> replot
```

Here is the plot resulting from this session.

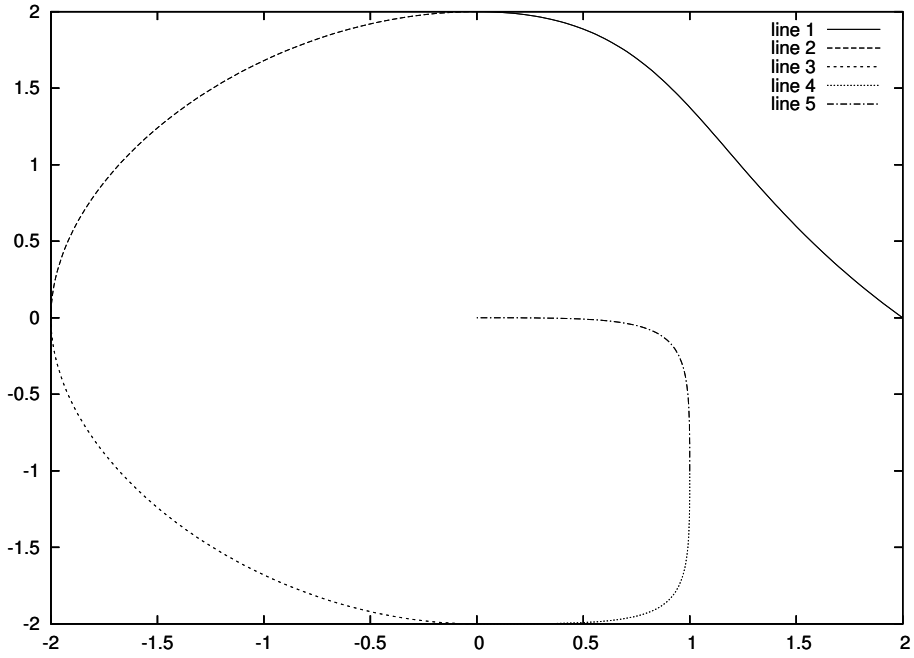


Here is the session I used to create the Bezier splines. In this session xh stores the x-coordinates of the control points and yh stores the y-coordinates. Since the formulae for the splines are in terms $\alpha = \Delta x$ and $\beta = \Delta y$, we must first solve for alp and bet.

```
octave:15> xh=[1;-1;-2;1;1;-1]
xh =

   1
  -1
  -2
   1
   1
  -1

octave:16> yh=[1;2;-1;-2;0;0]
yh =

   1
   2
  -1
  -2
   0
   0

octave:17> alp=xh-x
alp =

  -1
  -1
   0
   1
   0
```

```
  -1

octave:18> alp=xh-x
alp =

  -1
  -1
   0
   1
   0
  -1

octave:19> alp=xh-x
alp =

  -1
  -1
   0
   1
   0
  -1

octave:20> bet=yh-y
bet =

   1
   0
  -1
   0
   1
   0

octave:21> t=linspace(0,1)';
octave:22> bezx=zeros(100,5);
octave:23> bezy=zeros(100,5);
octave:24> for i=1:5
>bezx(:,i)=(2*(x(i)-x(i+1))+3*(alp(i)+alp(i+1))).*t.^3+(3*(x(i+1)-x(i))-3*(alp(i+1)+2*alp(i))).*t.^2
+3*alp(i).*t+x(i);
>bezy(:,i)=(2*(y(i)-y(i+1))+3*(bet(i)+bet(i+1))).*t.^3+(3*(y(i+1)-y(i))-3*(bet(i+1)+2*bet(i))).*t.^2
+3*bet(i).*t+y(i);
> end
octave:25> gset terminal x11
octave:26> hold off
octave:27> plot(bezx(:,1),bezy(:,1))
octave:28> hold on
octave:29> plot(bezx(:,2),bezy(:,2))
octave:30>
octave:30> plot(bezx(:,3),bezy(:,3))
octave:31> plot(bezx(:,4),bezy(:,4))
octave:32> plot(bezx(:,5),bezy(:,5))
octave:33> gset output "lab3-07b.eps"
octave:34> gset terminal postscript eps
octave:35> replot
octave:36> diary off
```

Here is graph of the spline produced from this session.



Note: The function divdifplot is available on the course webpage. Feel free to use it. You may want to take a look to see how it works.