Assigned: April 1, 2015
Due: Week of April 20, 2015

MTH 371: Group Project 3
Cubic Splines

GENERAL GROUP PROJECT GUIDELINES:

- Group project assignments should be a collaborative effort. All should participate in discussion and solution writing.
- Two weeks after the project is assigned, your group will meet with Dr. Vidden to discuss. All members must be present. Your grade will be determined at the end of the meeting.
- Each student should keep group project solutions in a dedicated notebook. Bring this notebook to your weekly meeting to discuss your findings. For coded solutions, bring a laptop to your weekly meeting. Have the laptop ready before the start of the meeting.

Cubic splines are used extensively in a wide range of applications from Boeing designing airplanes to Pixar creating animated films. Check the papers on D2L for some details on this process. With this project, you will see the implementation of cubic splines as a way to model everyday shapes.

1. Make a plot of one of your group member's hand with Scilab. To do this, start with

```
z=locate();
x=z(1,:)';
y=z(2,:)';
```

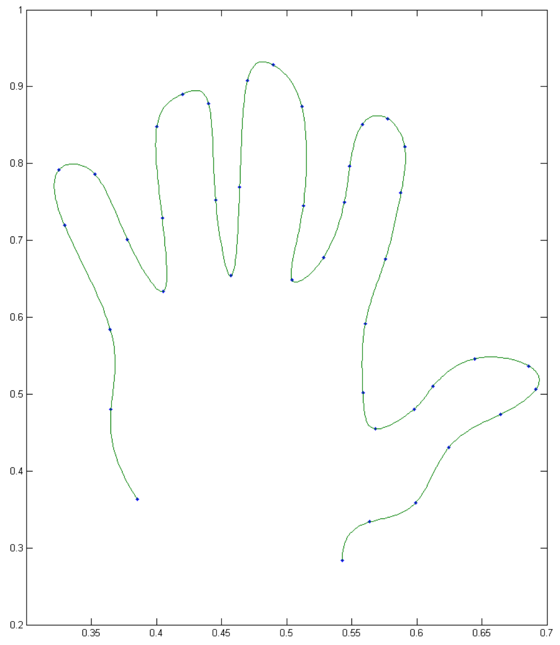
   The return is a sequence of points as you clicked them. Read the documentation for `locate()` for details. Use `plot(x,y)` to check your data.

2. Note, the curve which we will fit to these points is not a function. Hence, we create a parametric spline. Translate the attached Matlab function `splinetx(x,y,u)` to Scilab. Running the following script, create a spline-fitted outline of your hand. Make sure you completely understand this function you are rewriting and how parametric splines work. On the reverse page, I have include a resulting picture for my own hand!

```
n = length(x);
s = (1:n)';
t = (1:.05:n)';
u = splinetx(s,x,t);
v = splinetx(s,y,t);
xdel(winsid());
plot(x,y,'.',u,v,'-');
```

3. Use your handy data coupled with the built in Scilab function `splin`. Fit three types of splines to your data, not-a-knot, clamped, and natural. What differences do you see? Research these types of splines more to confirm the differences you see.

4. BONUS extra credit problems.
   (a) Rewrite the `splinetx` function to suit the clamped and natural spline conditions.
   (b) Devise your own artistic spline creation using the above methods.
   (c) Download the secret project 3 Scilab files from D2L and apply the above ideas to improve the result.



```
function v = splinetx(x,y,u)
%SPLINETX  Textbook spline function.
%  v = splinetx(x,y,u) finds the piecewise cubic
%  interpolatory spline S(x), with S(x(j)) = y(j),
%  and returns v(k) = S(u(k)).
%
%  See SPLINE, PCHIPTX.

%   Copyright 2012 Cleve Moler and The MathWorks, Inc.

%  First derivatives

   h = diff(x);
   delta = diff(y)./h;
   d = splineslopes(h,delta);

%  Piecewise polynomial coefficients

   n = length(x);
   c = (3*delta - 2*d(1:n-1) - d(2:n))./h;
   b = (d(1:n-1) - 2*delta + d(2:n))./h.^2;

%  Find subinterval indices k so that x(k) <= u < x(k+1)

   k = ones(size(u));
   for j = 2:n-1
      k(x(j) <= u) = j;
   end

%  Evaluate spline

   s = u - x(k);
   v = y(k) + s.*(d(k) + s.*(c(k) + s.*b(k)));
```

```
% ------------------------------------------------------

function d = splineslopes(h,delta)
%  SPLINESLOPES  Slopes for cubic spline interpolation.
%  splineslopes(h,delta) computes d(k) = S'(x(k)).
%  Uses not-a-knot end conditions.

%  Diagonals of tridiagonal system

   n = length(h)+1;
   a = zeros(size(h)); b = a; c = a; r = a;
   a(1:n-2) = h(2:n-1);
   a(n-1) = h(n-2)+h(n-1);
   b(1) = h(2);
   b(2:n-1) = 2*(h(2:n-1)+h(1:n-2));
   b(n) = h(n-2);
   c(1) = h(1)+h(2);
   c(2:n-1) = h(1:n-2);

%  Right-hand side

   r(1) = ((h(1)+2*c(1))*h(2)*delta(1)+ ...
          h(1)^2*delta(2))/c(1);
   r(2:n-1) = 3*(h(2:n-1).*delta(1:n-2)+ ...
              h(1:n-2).*delta(2:n-1));
   r(n) = (h(n-1)^2*delta(n-2)+ ...
          (2*a(n-1)+h(n-1))*h(n-2)*delta(n-1))/a(n-1);

%  Solve tridiagonal linear system

   d = tridisolve(a,b,c,r);
```