COMP 350 Solutions to Assignment 5, 2015

1. $x = [1\ 2\ 3\ 4]^\top$, $y = [2\ 0\ -10\ -34]^\top$

- Vandermonde Form
  Let $p(x) = c_0 + c_1x + c_2x^2 + c_3x^3$, then we can plug in the data and write in the form of $Ac = y$, so

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 \\ 1 & 3 & 9 & 27 \\ 1 & 4 & 16 & 64 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \\ -10 \\ -34 \end{bmatrix}$$

  Solve it and we get:

$$c = \begin{bmatrix} 2 \\ -1 \\ 2 \\ -1 \end{bmatrix}$$

  Thus $p(x) = 2 - x + 2x^2 - x^3$.

- Lagrange Form
  We can write:

$$p(x) = l_0(x)y_0 + l_1(x)y_1 + l_2(x)y_2 + l_3(x)y_3,$$

  where, we have:

$$l_0(x) = \frac{(x-x_1)(x-x_2)(x-x_3)}{(x_0-x_1)(x_0-x_2)(x_0-x_3)} = \frac{(x-2)(x-3)(x-4)}{(1-2)(1-3)(1-4)} = -\frac{x^3-9x^2+26x-24}{6}$$

$$l_1(x) = \frac{(x-x_0)(x-x_2)(x-x_3)}{(x_1-x_0)(x_1-x_2)(x_1-x_3)} = \frac{(x-1)(x-3)(x-4)}{(2-1)(2-3)(2-4)} = \frac{x^3-8x^2+19x-12}{2}$$

$$l_2(x) = \frac{(x-x_0)(x-x_1)(x-x_3)}{(x_2-x_0)(x_2-x_1)(x_2-x_3)} = \frac{(x-1)(x-2)(x-4)}{(3-1)(3-2)(3-4)} = -\frac{x^3-7x^2+14x-8}{2}$$

$$l_3(x) = \frac{(x-x_0)(x-x_1)(x-x_2)}{(x_3-x_0)(x_3-x_1)(x_3-x_2)} = \frac{(x-1)(x-2)(x-3)}{(4-1)(4-2)(4-3)} = \frac{x^3-6x^2+11x-6}{6}$$

  Thus,

$$\begin{aligned} p(x) =& 2 * (-\tfrac{x^3-9x^2+26x-24}{6}) + 0 * (\tfrac{x^3-8x^2+19x-12}{2}) + \\ & (-10) * (-\tfrac{x^3-7x^2+14x-8}{2}) + (-34) * (\tfrac{x^3-6x^2+11x-6}{6}) \\ =& 2 - x + 2x^2 - x^3 \end{aligned}$$

- Newton Form
  Use the following table and we get $a_0 = 2,\ a_1 = -2,\ a_2 = -4,\ a_3 = -1$.

1	2			
2	0	$\frac{0-2}{2-1} = -2$		
3	-10	$\frac{-10-0}{3-2} = -10$	$\frac{-10+2}{3-1} = -4$	
4	-34	$\frac{-34+10}{4-3} = -24$	$\frac{-24+10}{4-2} = -7$	$\frac{-7+4}{4-1} = -1$

  Thus,

$$p(x) = 2 + (-2)(x-1) + (-4)(x-1)(x-2) + (-1)(x-1)(x-2)(x-3) = 2 - x + 2x^2 - x^3$$

2. The x-coordinate of the 7 knots we choose are:

$$t = linspace(-1, 1, 7) = [-1 \quad -0.6667 \quad -0.3333 \quad 0 \quad 0.3333 \quad 0.6667 \quad 1]^\top.$$

Also, the following code is used to compute $f(x)$:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function y=q2_f(x)
    n=size(x,1);
    y=zeros(n,1);
    for i=1:n
        y(i)=1/(1+25*x(i)^2);
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

(a) Newton method:
Code for computing the parameters:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function a=newton(x,y)
    %Input:
    %   t,y are the vectors storing the knots.
    %Output:
    %   a is the parameters for constructing the newton interpolation.
    n=size(x,1);
    a=zeros(n,1);
    for i=1:n
        a(i)=y(i);
        for j=i+1:n
            y(j)=(y(j)-y(i))/(x(j)-x(i));
        end
    end
    a(n)=y(n);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Code for evaluation:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function result=eval_newton(x,a,t)
    %Input:
    %   x is a vector of points to be evaluated at.
    %   a are parameters computed from ls().
    %   t stores the x-coordinate of the knots.
    %Output:
    %   result are the IP values corresponding to x
    n=size(t,1);
    m=size(x,1);
    result=zeros(m,1);
    for i=1:m
```

```
        result(i)=a(n);
        for j=n-1:-1:1
            result(i)=result(i)*(x(i)-t(j))+a(j);
        end
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

(b) Cubic spline:
Code for computing the parameters:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [z,h]=cubic_spline(t,y)
    %Input:
    %   t,y are the vectors storing the knots.
    %Output:
    %   z,h are the parameters for constructing the cubic spline interpolation.
    n=size(t,1);
    z=zeros(n,1);
    h=zeros(n-1,1);
    b=zeros(n-1,1);
    u=zeros(n-1,1);
    v=zeros(n-1,1);
    for i=1:n-1
        h(i)=t(i+1)-t(i);
        b(i)=(y(i+1)-y(i))/h(i);
    end
    u(2)=2*(h(1)+h(2));
    v(2)=6*(b(2)-b(1));
    for i=3:n-1
        u(i)=2*(h(i-1)+h(i))-h(i-1)*h(i-1)/u(i-1);
        v(i)=6*(b(i)-b(i-1))-h(i-1)*v(i-1)/u(i-1);
    end
    z(n)=0;
    for i=n-1:-1:2
        z(i)=(v(i)-h(i)*z(i+1))/u(i);
    end
    z(1)=0;
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Code for evaluation:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function result=eval_cubic_spline(x,t,y,z,h)
    %Input:
    %   x is a vector of points to be evaluated at.
    %   t,y are the vectors storing the knots.
    %   z,h are parameters computed from cubic_spline().
    %Output:
    %   result are the IP values corresponding to x
```

```
    n=size(t,1);
    A=zeros(n,1);
    B=zeros(n,1);
    C=zeros(n,1);
    D=zeros(n,1);
    for i=1:n-1
        A(i)=y(i);
        B(i)=h(i)*z(i+1)/(-6)-h(i)*z(i)/3+(y(i+1)-y(i))/h(i);
        C(i)=z(i)/2;
        D(i)=(z(i+1)-z(i))/(6*h(i));
    end

    m=size(x,1);
    result=zeros(m,1);
    for j=1:m
        for i=1:n-1
            if t(i)<=x(j)&&x(j)<=t(i+1)
                idx=i;
                break
            end
        end
        result(j)=A(idx)+(x(j)-t(idx))*(B(idx)+(x(j)-t(idx))*(C(idx)+(x(j)-t(idx))*D(idx)));
    end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

(c) Least squares:
The result interpolation expression:

$$g(x) = 0.7192 - 2.3862x^2 + 1.7195x^4$$

Code for computing the parameters:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function c=ls(t,y)
    %Input:
    %   t,y are the vectors storing the knots.
    %Output:
    %   c are the parameters for constructing the least squares interpolation.
    n=size(t,1);
    A=ones(n,3);
    for i=1:n
        A(i,2)=t(i)^2;
        A(i,3)=A(i,2)^2;
    end
    c=(A'*A)\(A'*y);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Code for evaluation:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
    function result=eval_ls(x,c)
        %Input:
        %   x is a vector of points to be evaluated at.
        %   c are parameters computed from ls().
        %Output:
        %   result are the IP values corresponding to x
        n=size(x,1);
        result=zeros(n,1);
        for i=1:n
            result(i)=c(1)+c(2)*x(i)^2+c(3)*x(i)^4;
        end
    end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

The result of $f(x),\ f(x)-p(x),\ f(x)-S(x),\ f(x)-g(x)$:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
      x        f(x)         f(x)-p(x)        f(x)-S(x)       f(x)-g(x)
-----------------------------------------------------------------
 -1.000000 3.846154e-02    0.000000e+00    0.000000e+00   -1.403975e-02
 -0.833333 5.446293e-02   -5.534161e-01   -1.867819e-02    1.631045e-01
 -0.666667 8.256881e-02   -1.387779e-17   -1.387779e-17    8.423848e-02
 -0.500000 1.379310e-01    2.293468e-01    5.393273e-02   -9.219636e-02
 -0.333333 2.647059e-01    0.000000e+00   -5.551115e-17   -2.105962e-01
 -0.166667 5.901639e-01   -1.817228e-01   -1.289242e-01   -6.408512e-02
  0.000000 1.000000e+00    0.000000e+00    0.000000e+00    2.807949e-01
  0.166667 5.901639e-01   -1.817228e-01   -1.289242e-01   -6.408512e-02
  0.333333 2.647059e-01    1.665335e-16    0.000000e+00   -2.105962e-01
  0.500000 1.379310e-01    2.293468e-01    5.393273e-02   -9.219636e-02
  0.666667 8.256881e-02   -2.914335e-16   -6.938894e-17    8.423848e-02
  0.833333 5.446293e-02   -5.534161e-01   -1.867819e-02    1.631045e-01
  1.000000 3.846154e-02    6.605827e-15    1.387779e-17   -1.403975e-02
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

The code for generating the above output:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function print_values()
  x=linspace(-1,1,13);
  x=x';
  f=q2_f(x);
  t=linspace(-1,1,7);
  t=t';
  y=q2_f(t);

  a=newton(t,y);
  result_nt=eval_newton(x,a,t);
  [z,h]=cubic_spline(t,y);
  result_cs=eval_cubic_spline(x,t,y,z,h);
```

```
  c=ls(t,y);
  result_ls=eval_ls(x,c);
  disp('      x        f(x)         f(x)-p(x)        f(x)-S(x)       f(x)-g(x)');
  disp('-------------------------------------------------------------');
  for i=1:13
    disp(sprintf('%10.6f %10.6e %15.6e %15.6e %15.6e', x(i), f(i), ...
       f(i)-result_nt(i), f(i)-result_cs(i), f(i)-result_ls(i)))
  end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

The plot of the three interpolations is shown in Fig. 1. The code for plotting:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function q2_plot(x,t,y)
    %Input:
    %   x is a vector of points to be evaluated at.
    %   t,y are the vectors storing the knots.

    %newton
    a=newton(t,y);
    result_nt=eval_newton(x,a,t);
    %cubic spline
    [z,h]=cubic_spline(t,y);
    result_cs=eval_cubic_spline(x,t,y,z,h);
    %least squares
    c=ls(t,y);
    result_ls=eval_ls(x,c);

    h=plot(x,q2_f(x),'k^-',x,result_nt,'r-o',x,result_cs,'g-+',x,result_ls,'b-*');
    h_leg=legend('original','newton','cubic spline','least squares');
    set(h,{'markers'},{14;10;10;10});
    set(h_leg,'FontSize',14);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

3. Code and output:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
figure('position',get(0,'screensize'));
axes('position',[0 0 1 1]);
[x,y]=ginput;
for i=1:length(x)
    xy(1,i)=x(i);
    xy(2,i)=y(i);
end;
t=1:length(x);
ts=1:0.1:length(x);
```

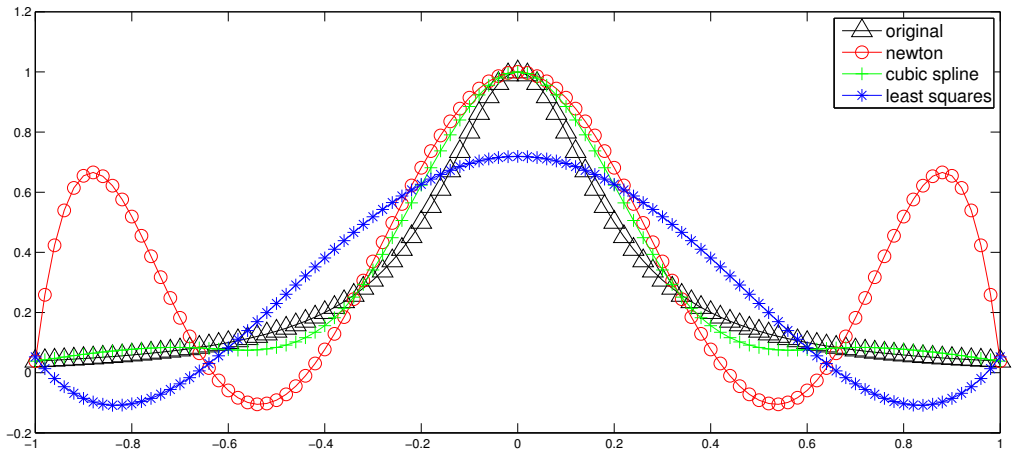

Figure 1: Plot of the three kinds of interpolation. $x = [-1 : 0.02 : 1]$

```
xys=spline(t,xy,ts);
plot(xys(1,:),xys(2,:));
disp('    x    y');
disp('  ---------');
disp(xy');
```

```
>     x          y
  -------------------
    0.3052    0.0604
    0.2417    0.2294
    0.2173    0.3161
    0.1616    0.4439
    0.1069    0.5930
    0.1206    0.6371
    0.1440    0.6342
    0.1821    0.5391
    0.2700    0.4126
    0.2036    0.7337
    0.1899    0.7862
    0.2065    0.8359
    0.2446    0.8232
    0.2603    0.7152
    0.2935    0.6158
    0.3325    0.5078
    0.3110    0.6754
    0.2954    0.8161
    0.2886    0.8899
    0.3140    0.9268
    0.3384    0.9098
    0.3462    0.8274
    0.3794    0.7010
    0.3950    0.5376
```

```
    0.4155    0.5362
    0.4253    0.6357
    0.4370    0.7450
    0.4458    0.8601
    0.4761    0.9112
    0.5015    0.8928
    0.5015    0.7706
    0.5005    0.6598
    0.4888    0.5376
    0.5015    0.3999
    0.5308    0.3928
    0.5522    0.4879
    0.5610    0.5888
    0.5903    0.6243
    0.6226    0.6115
    0.6284    0.4794
    0.6069    0.3388
    0.5601    0.0717
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

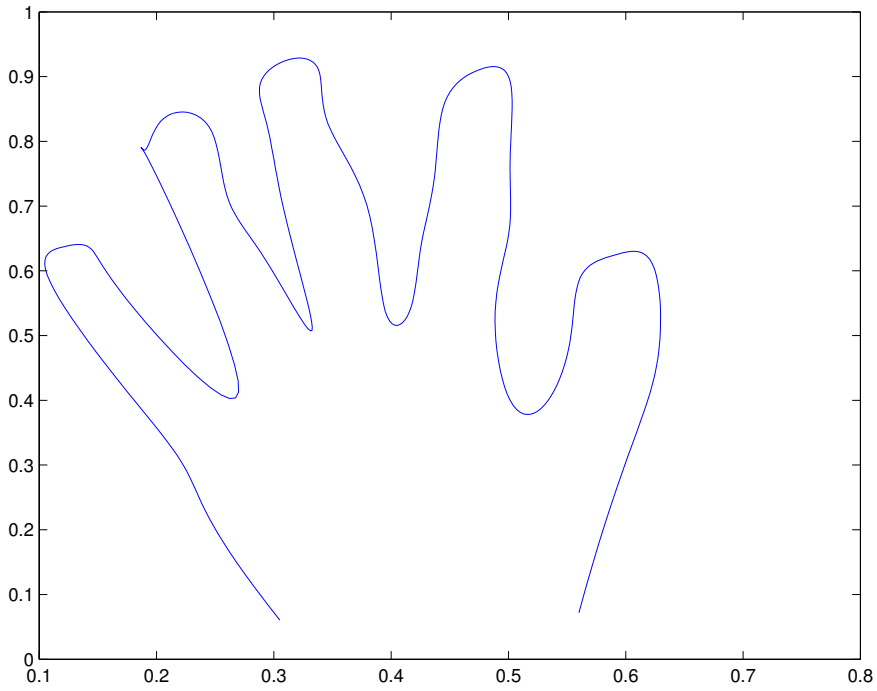

Figure 2: Hand