Numerical Analysis MTH614

Spring 2012, Korea University

Interpolations

Interpolations

For n discrete data points, $\{(x_i, y_i)|i = 1, 2, \cdots, n\}$

$$p_n(x_1) = y_1, \quad p_n = y_2, \cdots, p_n(x_n) = y_n,$$

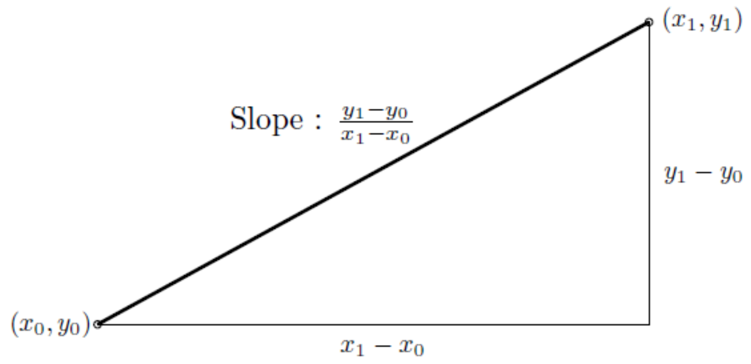
there is one and only one polynomial of order $n-1$ that passes through all the points. Polynomial interpolation consists of determining the unique the order $n-1$ polynomial that fits n data points.

$$f(x) = a_1 + a_2 x + a_3 x^2 + \cdots + a_n x^{n-1}.$$

This polynomial then provides an interpolating equation to compute intermediate values.

Linear Interpolation

The simplest form of interpolation is to connect two data points with a straight line. This is called linear interpolation is described as follows



Using this triangle, we can deduce the equation,

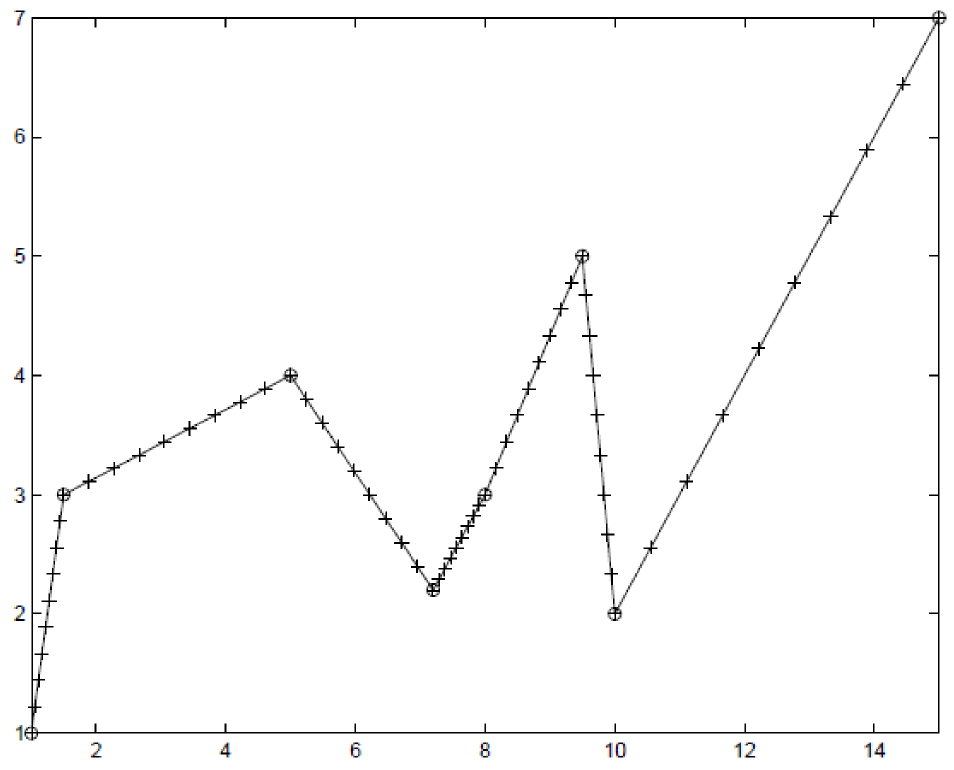
$$P_1(x) = y_0 + \frac{y_1 - y_0}{x_1 - x_0}(x - x_0) = \frac{x - x_1}{x_0 - x_1}y_0 + \frac{x - x_0}{x_1 - x_0}y_1$$

which is the Newton linear interpolation formula.

For example, let us have distinct eight points. Then plot a series of line segments connecting all your data using piecewise linear interpolation.

MATLAB CODE

```
%%%%%%%%%%%%%%%%%%%%%%% lininterp.m %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clc; clear; clf;
xp=[1 1.5 5 7.2 8 9.5 10 15];
yp=[1 3 4 2.2 3 5 2 7];
plot(xp,yp,'ko'); hold
np=length(xp); n=10;
for i=1:np-1
   x=linspace(xp(i+1),xp(i),n);
   y=(yp(i+1)-yp(i))/(xp(i+1)-xp(i))*(x-xp(i))+yp(i);
plot(x,y,'k+-');
end
axis([xp(1) xp(np) min(yp) max(yp)]);
```



1
2
3
4
5
6
7
2
4
6
8
10
12
14

Cubic splines

Cubic splines are also most frequently used in practice. This is because cubic splines provide the simplest representation that exhibits the smoothing curves of the function.

This method is to derive a third-order polynomial for each interval as represented by

$$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3.$$

satisfying the followings in subinterval $t_i \le x \le t_{i+1}$.

$$S(x) = \begin{cases} S_1(x), & t_1 \le x \le t_2 \\ S_2(x), & t_2 \le x \le t_3 \\ \vdots & \vdots \\ S_{n-1}(x), & t_{n-1} \le x \le t_n \end{cases}$$

This means that each of the cubics must join at the knots.

Next, second derivatives at the interior nodes must be equal. We let $z_i = S''(t_i)$, and its end points have the condition of $z_1 = z_n = 0$

$$S_i''(x) = \frac{z_{i+1}}{h_i}(x - t_i) + \frac{z_i}{h_i}(t_{i+1} - x) \tag{1}$$

where h_i is the size of the i th subinterval $h_i = t_{i+1} - t_i$.

Integrate twice,

$$S_i(x) = \frac{z_{i+1}}{6h_i}(x - t_i)^3 + \frac{z_i}{6h_i}(t_{i+1} - x)^3 + a(x - t_i) + b(t_{i+1} - x) \tag{2}$$

Since $S_i(t_i) = y_i,\ S_i(t_{i+1}) = y_{i+1}$,

$$a_i = \frac{y_{i+1}}{h_i} - \frac{h_i}{6}z_{i+1}, \quad b_i = \frac{y_i}{h_i} - \frac{h_i}{6}z_i. \tag{3}$$

And we know $z_1 = z_n = 0$ with undetermined variables $z_2, \ldots, z_{n-1}$, we have

$$\begin{aligned} S_i(x) \quad = \quad & \frac{z_{i+1}}{6h_i}(x - t_i)^3 + \frac{z_i}{6h_i}(t_{i+1} - x)^3 \\ & + \left(\frac{y_{i+1}}{h_i} - \frac{h_i}{6}z_{i+1}\right)(x - t_i) + \left(\frac{y_i}{h_i} - \frac{h_i}{6}z_i\right)(t_{i+1} - x) \,. \end{aligned} \tag{4}$$

The undetermined variables are found by equating the first derivatives at the interior nodes,

$$\begin{aligned} S_i'(x) &= \frac{z_{i+1}}{2h_i}(x-t_i)^2 - \frac{z_i}{2h_i}(t_{i+1}-x)^2 \\ &\quad + \left(\frac{y_{i+1}}{h_i} - \frac{h_i}{6}z_{i+1}\right) - \left(\frac{y_i}{h_i} - \frac{h_i}{6}z_i\right). \end{aligned} \tag{1}$$

And again we let $c_i = (y_{i+1}-y_i)/h_i$,

$$\begin{aligned} S_i'(t_i) &= -\frac{h_i}{2}z_i + \left(\frac{y_{i+1}}{h_i} - \frac{h_i}{6}z_{i+1}\right) - \left(\frac{y_i}{h_i} - \frac{h_i}{6}z_i\right) \qquad (2) \\ &= -\frac{h_i}{6}z_{i+1} - \frac{h_i}{3}z_i + c_i \\ S_{i-1}'(t_i) &= \frac{h_{i-1}}{3}z_i + \frac{h_{i-1}}{6}z_{i-1} + c_{i-1} \quad . \qquad (3) \end{aligned}$$

Set up to $S_i'(t_i) = S_{i-1}'(t_i)$ for $i = 2, \ldots, n-1$,

$$h_{i-1}z_{i-1} + 2(h_{i-1}+h_i)z_i + h_i z_{i+1} = 6(c_i - c_{i-1}). \tag{4}$$

To solve the system of equations, we define

$$u_i = 2(h_{i-1} + h_i),\ v_i = 6(c_i - c_{i-1}) \tag{1}$$

and

$$\begin{cases} z_1 = 0 \\ h_{i-1}z_{i-1} + u_i z_i + h_i z_{i+1} = v_i, \quad 2 \le i \le n-1 \\ z_n = 0 \end{cases} \tag{2}$$

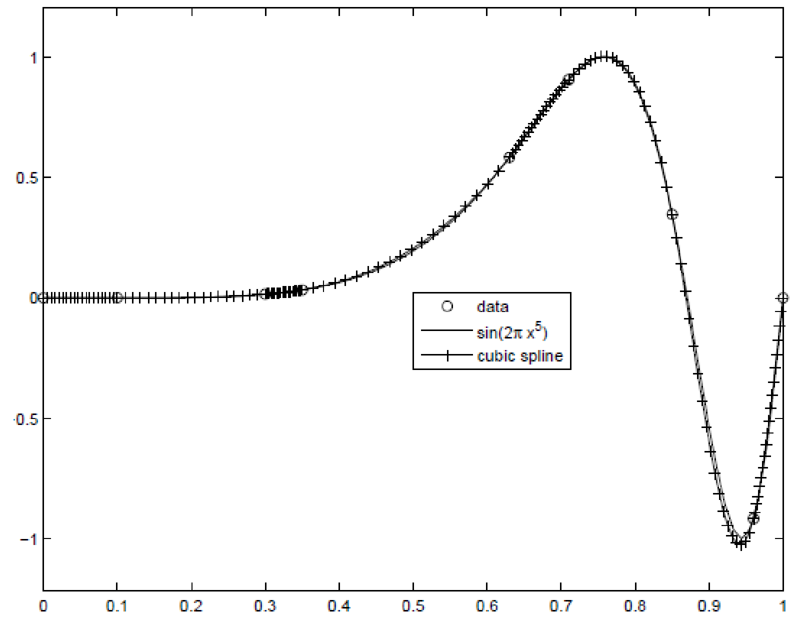
then

$$\begin{pmatrix} u_2 & h_2 & 0 & \cdots & \cdots & 0 \\ h_2 & u_3 & h_3 & 0 & \cdots & 0 \\ 0 & h_3 & u_4 & \ddots & \cdots & \vdots \\ \vdots & 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \vdots & \ddots & \ddots & u_{n-2} & h_{n-2} \\ 0 & 0 & \cdots & 0 & h_{n-2} & u_{n-1} \end{pmatrix} \begin{pmatrix} z_2 \\ z_3 \\ \vdots \\ \vdots \\ z_{n-2} \\ z_{n-1} \end{pmatrix} = \begin{pmatrix} v_2 \\ v_3 \\ \vdots \\ \vdots \\ v_{n-2} \\ v_{n-1} \end{pmatrix} \tag{3}$$

For example, let us plot cubic spline interpolation. We assign some points in x axis and then evaluate them with given sine function at some points. After that, we reconstruct the function with given data set.

```
%%%%%%%%%%%%%%%%%%%%%%%% natural_cubic.m %%%%%%%%%%%%%%%%%%%%%%%%
clear; clc; clf; f=inline('sin(2*pi*x.^5)');
x=[0 0.1 0.3 0.35 0.63 0.71 0.85 0.96 1];
y=f(x); n=length(x); xp=linspace(x(1),x(n),10*n);
plot(x,y,'ko'); hold
plot(xp,f(xp),'k-');
for i=1:n-1
 h(i)=x(i+1)-x(i); c(i)=(y(i+1)-y(i))/h(i);
end
A=zeros(n-2);
for i=1:n-2
  A(i,i)=2*(h(i+1)+h(i)); v(i)=6*(c(i+1)-c(i));
end
for i=2:n-2
 A(i,i-1)=h(i); A(i-1,i)=h(i);
end
```

```
z(1)=0; z(n)=0;            % Natural spline
z(2:n-1)=inv(A)*v';
for i=1:n-1
    t=linspace(x(i),x(i+1),20);
  S=z(i+1)/(6*h(i))*(t-x(i)).^3+z(i)/(6*h(i))*(x(i+1)-t).^3+...
      (y(i+1)/h(i)-h(i)/6*z(i+1))*(t-x(i))+...
      (y(i)/h(i)-h(i)/6*z(i))*(x(i+1)-t);
plot(t,S,'k+-');
end
axis([x(1) x(n) min(y)-0.3 max(y)+0.3])
legend('data','sin(2\pi x^5)','cubic spline')
```


data
sin(2π x^5)
cubic spline