Interpolation and Approximation Theory

Finding a polynomial of at most degree n to pass through $n+1$ points in the interval $[a, b]$ is referred to as "*interpolation*".

Approximation theory deals with two types of problems.

- Given a data set, one seeks a function best fitted to this data set, for example, given $\{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}$, one seeks a line $y = mx + b$ which best fits this data set.
- Given an explicit function, one seeks a simpler function for representation, for example, use $1 + x + \frac{x^2}{2!} + \frac{x^3}{3!}$ to represent e^x.

♣ Lagrange Polynomial Interpolation

♣ Newton's Divided-Difference Formula

♡ Hermite Polynomial Interpolation

♣ Cubic spline interpolation

♣ Bezier curves

♣ Cubic B-splines

♣ Orthogonal functions

♣ Trigonometric functions

♣ Chebyshev polynomials

♡ Legendre polynomials

♡ Laguerre polynomials

♡ Gamma functions

♡ Beta functions

♡ Bessel functions

♢ Other Topics with Applications

Polynomial Approximation

Suppose that the function $f(x) = e^x$ is to be approximated by a polynomial of degree 2 over the interval [-1, 1]. The approximations by Taylor polynomial $1 + x + 0.5x^2$ and Chebyshev polynomial $1 + 1.17518x + 0.54309x^2$ are given below.

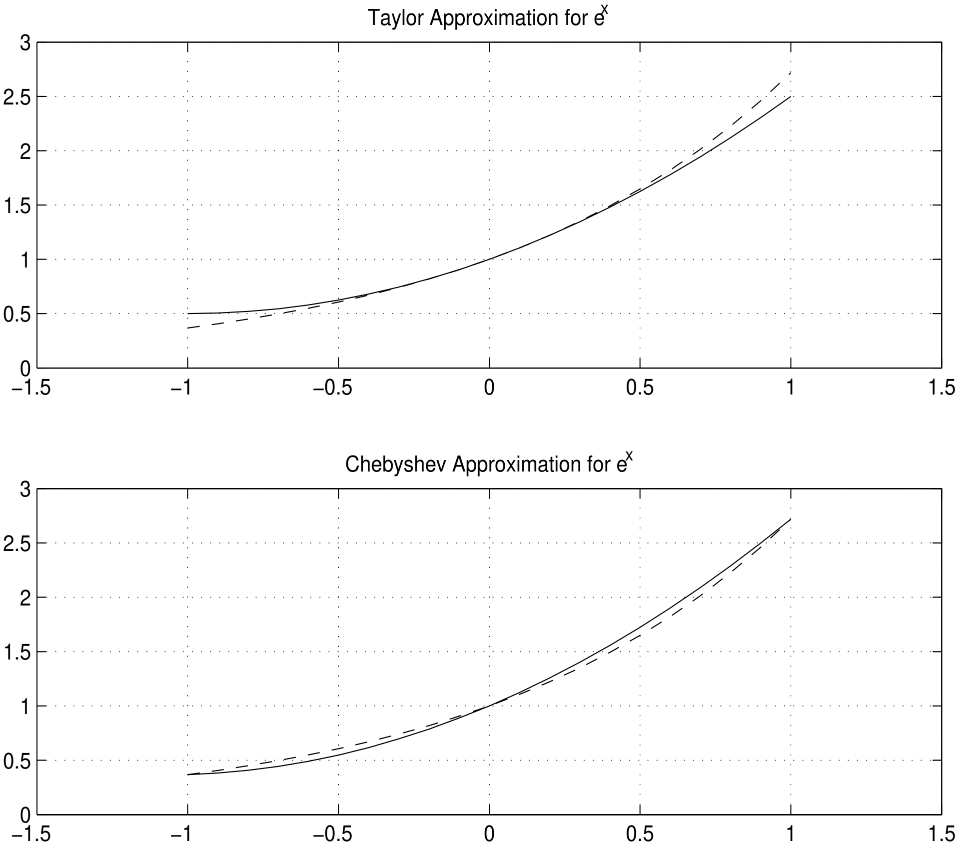

Figure 1: Polynomial Approximations for e^x over [-1, 1]

```
X=-1:0.1:1;
Y0=exp(X);
Y1=1.0000+1.0000*X+0.5000*X.^2;      % Taylor Expansion
Y2=1.0000+1.17518*X+0.54309*X.^2;    % Chebyshev Polynomial by Chaurchin
%Y2=1.0000+1.129772*X+0.532042*X.^2; % Chebyshev Polynomial from Textbook
V=[-1.5 1.5, 0 3];
subplot(2,1,1)
plot(X,Y0,'b--',X,Y1,'r-'); axis(V); grid;
title('Taylor Approximation for e^x')
subplot(2,1,2)
plot(X,Y0,'b--',X,Y2,'r-'); axis(V); grid;
title('Chebyshev Approximation for e^x')
```

Taylor Polynomial Approximation

Suppose that $f \in C^{n+1}[a,b]$ and $x_0 \in [a,b]$ is a fixed value. If $x \in [a,b]$, then

$$f(x) = P_n(x) + E_n(x)$$

where $P_n(x)$ is a polynomial that can be used to approximate $f(x)$ by

$$f(x) \approx P_n(x) = \sum_{k=0}^{n} \frac{f^{(k)}(x_0)}{k!}(x-x_0)^k$$

having some c between x and x_0 such that

$$E_n(x) = \frac{f^{(n+1)}(c)}{(n+1)!}(x-x_0)^{n+1}$$

- $|e - P_{15}(1)| = |e - 2.718282818459| < \frac{e}{16!} < \frac{3}{16!} < 1.433844 \times 10^{-13}$
- $|sin(x) - P_9(x)| < \frac{1}{10!} \leq 2.75574 \times 10^{-7}$ for $|x| \leq 1$, where

$$P_9(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!}$$

- $|cos(x) - P_8(x)| < \frac{1}{9!} \leq 2.75574 \times 10^{-6}$ for $|x| \leq 1$, where

$$P_8(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!}$$

Polynomial Interpolation

We attempt to find a polynomial of at most degree n to pass through $n+1$ points in the interval $[a, b]$.

$$[x_0, y_0]^t,\ [x_1, y_1]^t,\ \cdots,\ [x_n, y_n]^t$$

where

$$a\ \le\ x_0\ <\ x_1\ <\ \cdots\ <\ x_n\ \le b$$

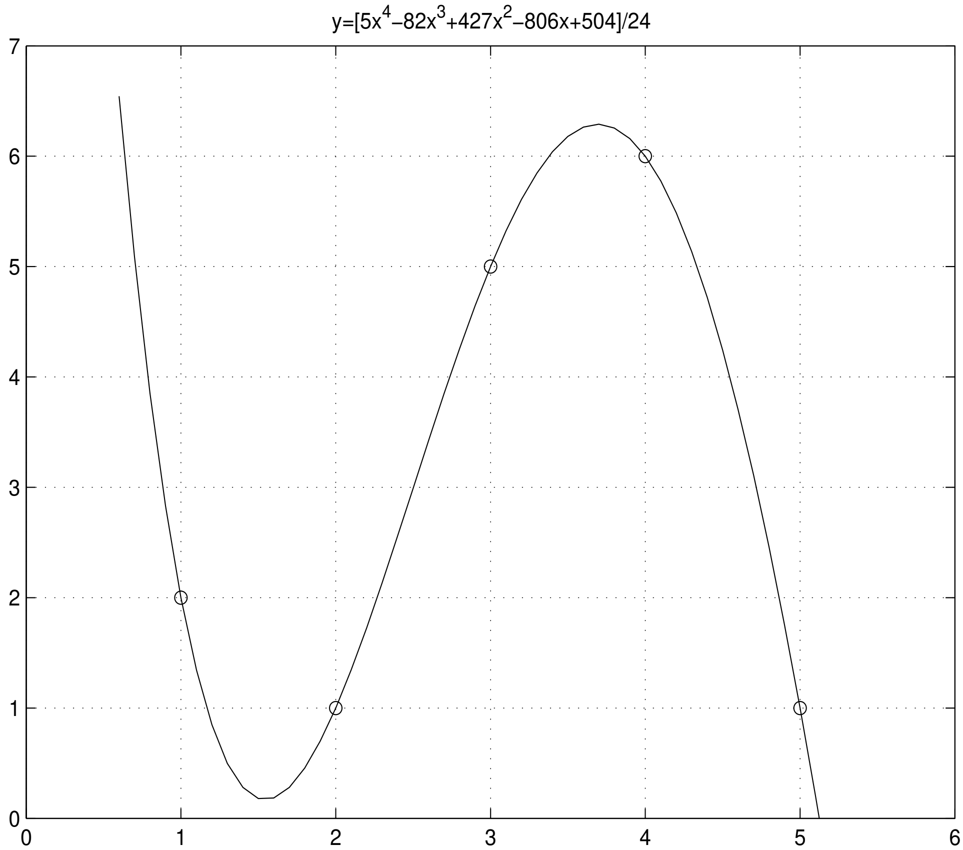

Figure 2: Polynomial Passing Through Five Points

```
%
% Script File: func4.m
% A quadric function for interpolation: y=f(x)=[5x^4-82x^3+427x^2-806x+504]/24
%
X=0.6:0.1:5.2;
Y=(5*X.^4-82*X.^3+427*X.^2-806*X+504)/24.0;
V=[0 6, 0 7];
plot(X,Y,'b-',[1 2 3 4 5],[2 1 5 6 1],'ro'); axis(V); grid
title('y=[5x^4-82x^3+427x^2-806x+504]/24')
```

Polynomials for Interpolation

Theorem: Suppose that the function $y = f(x)$ is known at the $n+1$ distinct points

$$[x_0, y_0]^t,\ [x_1, y_1]^t,\ \cdots,\ [x_n, y_n]^t$$

where

$$a \ \leq \ x_0 \ < \ x_1 \ < \ \cdots \ < \ x_n \ \leq b$$

Then there is a unique polynomial $P_n(x)$ of degree at most n such that

$$P_n(x_i) = y_i \ \ \forall \ \ 0 \leq i \leq n$$

If the error function $E(x) = f(x) - P_n(x)$ is required, then we need to know $f^{(n+1)}(x)$ whose bound of magnitude is

$$max\{|f^{(n+1)}(x)| : \ a \leq x \leq b\}$$

• A Lagrange polynomial of degree n

$$L_{n,k}(x) = \frac{\prod_{j\neq k}^{n}(x - x_j)}{\prod_{j\neq k}^{n}(x_k - x_j)}$$

♡ *Error Formula for Lagrange Polynomial*

$$f(x) = \sum_{k=0}^{n} f(x_k) L_{n,k}(x) + \frac{f^{(n+1)}(\xi_x)}{(n+1)!} \prod_{k=0}^{n} (x - x_k)$$

for some unknown number ξ_x that lies in the smallest interval that contains $x_0,\ x_1,\ \cdots,\ x_n$, and x.

• Polynomials in Newton Form

$$P_n(x) = P_{n-1}(x) + a_n \prod_{j=0}^{n-1} (x - x_j)$$

• Polynomials in Chebyshev Form

$$P_n(x) = \alpha_0 + \alpha_1 T_1(x) + \alpha_2 T_2(x) + \cdots + \alpha_n T_n(x)$$

where

$$T_n(x) = cos(ncos^{-1}x), \ \ T_0(x) \equiv 1,\ T_1(x) = x,\ T_2(x) = 2x^2 - 1,\ T_3(x) = 4x^3 - 3x.$$

♠ Hermite Polynomials $H_n(x)$

An Example for Polynomial Interpolation

We look for polynomials of degree at most 3 to interpolate the following *four* points.

x	5	-7	-6	0
y	1	-23	-54	-954

Table 1: $P_3(x) = 4x^3 + 35x^2 - 84x - 954$

.

♡ Solution in Lagrange form

$$\begin{aligned} P_3(x) &= 1 \cdot \tfrac{(x+7)(x+6)(x-0)}{(5+7)(5+6)(5-0)} \\ &+ (-23) \cdot \tfrac{(x-5)(x+6)(x-0)}{(-7-5)(-7+6)(-7-0)} \\ &+ (-54) \cdot \tfrac{(x-5)(x+7)(x-0)}{(-6-5)(-6+7)(-6-0)} \\ &+ (-954) \cdot \tfrac{(x-5)(x+7)(x-6)}{(0-5)(0+7)(0+6)} \end{aligned}$$

♡ Solution in Newton form

$$P_3(x) = 1 + 2(x-5) + 3(x-5)(x+7) + 4(x-5)(x+7)(x+6)$$

♡ Solution in Chebyshev form

$$P_3(x) = -936.5 - 81T_1(x) + 17.5T_2(x) + T_3(x)$$

where

$$T_n(x) = cos(ncos^{-1}x), \quad T_0(x) \equiv 1, \ T_1(x) = x, \ T_2(x) = 2x^2 - 1, \ T_3(x) = 4x^3 - 3x.$$

Divided Differences

Suppose that the function $y = f(x)$ is known at the $n+1$ points

$$[x_0, f(x_0)]^t,\ [x_1, f(x_1)]^t,\ \cdots,\ [x_n, f(x_n)]^t,\ \ \textit{where}\ \ a \le x_0 < x_1 < \cdots < x_n \le b$$

The $n+1$ zeroth divided differences of f are defined as

$$f[x_i] = f(x_i)\ 0 \le i \le n$$

The first divided differences of f are defined as

$$f[x_i, x_{i+1}] = \frac{f[x_{i+1}] - f[x_i]}{x_{i+1} - x_i}\ \ \forall\ 0 \le i \le n-1$$

The kth divided differences can be inductively defined by

$$f[x_i, x_{i+1}, \cdots, x_{i+k-1}, x_{i+k}] = \frac{f[x_{i+1}, x_{i+2}, \cdots, x_{i+k}] - f[x_i, x_{i+1}, \cdots, x_{i+k-1}]}{x_{i+k} - x_i}\ \forall\ 0 \le i \le n-k$$

The nth divided difference is

$$f[x_0, x_1, \cdots, x_n] = \frac{f[x_1, x_2, \cdots, x_n] - f[x_0, x_1, \cdots, x_{n-1}]}{x_n - x_0}$$

It can be shown that the nth Lagrange interpolation polynomial w.r.t. $x_0 < x_1 < \cdots < x_n$ can be expressed as Newton (interpolatory) divided-difference formula

$$\begin{aligned} P_n(x) &= f[x_0] + f[x_0, x_1](x - x_0) + f[x_0, x_1, \cdots, x_n](x - x_0)(x - x_1)\cdots(x - x_{n-1}) \\ &= f[x_0] + \textstyle\sum_{k=1}^{n} f[x_0, x_1, \cdots, x_k](x - x_0)(x - x_1)\cdots(x - x_{k-1}) \end{aligned} \quad (1)$$

Newton (interpolatory) divided-difference formula has simpler form when $x_j - x_{j-1} = h\ \ \forall\ 1 \le j \le n$. Let $x = x_0 + sh$, then $x - x_i = (s-i)h$, then the formula **??** becomes

$$\begin{aligned} P_n(x) &= P_n(x_0 + sh) = f[x_0] + \textstyle\sum_{k=1}^{n} s(s-1)\cdots(s-k+1)h^k f[x_0, x-1, \cdots, x_k] \\ &= f[x_0] + \textstyle\sum_{k=1}^{n} \begin{pmatrix} s \\ k \end{pmatrix} k! h^k f[x_0, x_1, \cdots, x_k] \\ &= f[x_0] + \textstyle\sum_{k=1}^{n} \begin{pmatrix} s \\ k \end{pmatrix} \triangle^k f(x_0) \\ &= f[x_n] + \textstyle\sum_{k=1}^{n} (-1)^k \begin{pmatrix} -s \\ k \end{pmatrix} \nabla^k f(x_n) \end{aligned}$$

Hermite Interpolation and Polynomial

If $f \in C^1[a,b]$ and $a \leq x_0 < x_1 < \cdots < x_n \leq b$, the unique polynomial of least degree which agrees with f and f' at $x_0, x_1, \cdots, x_n$ is the polynomial of degree at most $2n+1$ given by

$$H_{2n+1}(x) = \sum_{j=0}^{n} f(x_j) H_{n,j} + \sum_{j=0}^{n} f'(x_j) \hat{H}_{n,j}(x)$$

where

$$H_{n,j}(x) = [1 - 2(x - x_j) L'_{n,j}(x_j)] L^2_{n,j}(x)$$

$$\hat{H}_{n,j}(x) = (x - x_j) L^2_{n,j}(x_j)$$

$$L_{n,j}(x) = \frac{(x - x_0)(x - x_1)\cdots(x - x_{j-1})(x - x_{j+1})\cdots(x - x_n)}{(x_j - x_0)(x_j - x_1)\cdots(x_j - x_{j-1})(x_j - x_{j+1})\cdots(x_j - x_n)}$$

- Show that $H_{2n+1}(x_k) = f(x_k)$ and $H'_{2n+1}(x_k) = f'(x_k) \;\; \forall\, k = 0, 1, \cdots, n.$
- *Error Formula*
  If $f \in C^{2n+2}[a,b]$, then

$$f(x) = H_{2n+1}(x) + \frac{f^{(2n+2)}(\xi_x)}{(2n+2)!}(x - x_0)^2 (x - x_1)^2 \cdots (x - x_n)^2$$

  for some $\xi_x \in (a,b)$.

Cubic Spline Interpolation

Given a function f defined on $[a,b]$ and a set of $n+1$ nodes $a = x_0 < x_1 < \cdots < x_n = b$, a cubic spline interpolant, S, for f is a function that satisfies the following conditions:

(1) For each $j = 0, 1, \cdots, n-1$, $S(x)$ is a cubic polynomial, denoted by $S_j(x)$, on the subinterval $[x_j, x_{j+1})$.

(2) $S(x_j) = f(x_j)$ for each $j = 0, 1, \cdots, n$.

(3) $S_{j+1}(x_{j+1}) = S_j(x_{j+1})$ for each $j = 0, 1, \cdots, n-2$.

(4) $S'_{j+1}(x_{j+1}) = S'_j(x_{j+1})$ for each $j = 0, 1, \cdots, n-2$.

(5) $S''_{j+1}(x_{j+1}) = S''_j(x_{j+1})$ for each $j = 0, 1, \cdots, n-2$.

(6) One of the following sets of boundary conditions is satisfied:

(a) $S''(x_0) = S''(x_n) = 0$ (natural or free boundary);

(b) $S'(x_0) = f'(x_0)$ and $S'(x_n) = f'(x_n)$ (clamped boundary).

x	0.9	1.3	1.9	2.1	2.6	3.0	3.9	4.4	4.7	5.0	6.0
$f(x)$	1.3	1.5	1.85	2.1	2.6	2.7	2.4	2.15	2.05	2.1	2.25
x	7.0	8.0	9.2	10.5	11.3	11.6	12.0	12.6	13.0	13.3	
$f(x)$	2.3	2.25	1.95	1.4	0.9	0.7	0.6	0.5	0.4	0.25	

Table 2: **A ruddy duck in flight**

Finding A Cubic Spline Interpolant

Let $S_j(x) = a_j + b_j(x - x_j) + c_j(x - x_j)^2 + d_j(x - x_j)^3$, $h_j = x_{j+1} - x_j$, for $0 \le j \le n-1$,

From (2), $a_j = S_j(x_j) = f(x_j)$, $0 \le j \le n-1$, and denote $a_n = f(x_n)$.
From (3), $a_{j+1} = a_j + b_j(x_{j+1} - x_j) + c_j(x_{j+1} - x_j)^2 + d_j(x_{j+1} - x_j)^3$, $0 \le j \le n-2$.

(**A**) $a_{j+1} = a_j + b_j h_j + c_j h_j^2 + d_j h_j^3$, $0 \le j \le n-1$, where $a_n = f(x_n)$.
Similarly, $S_j'(x) = b_j + 2c_j(x - x_j) + 3d_j(x - x_j)^2$, $0 \le j \le n-1$.

(**B**) $b_{j+1} = b_j + 2c_j h_j + 3d_j h_j^2$, $0 \le j \le n-1$ by (4).

Define $c_n = \frac{1}{2}S''(x_n)$, *and by using* (5), we have
(**C**) $c_{j+1} = c_j + 3d_j h_j$, $0 \le j \le n-1$, and $c_{n-1} + 3d_{n-1}h_{n-1} = c_n = 0$ by using (6)(a).

(**C′**) $d_j = \frac{1}{3h_j}(c_{j+1} - c_j)$, $0 \le j \le n-1$, substitute (**C′**) into (**A**) and (**B**), we have

(**D**) $a_{j+1} = a_j + b_j h_j + \frac{h_j^2}{3}(2c_j + c_{j+1})$, $0 \le j \le n-1$

(**E**) $b_{j+1} = b_j + h_j(c_j + c_{j+1})$, $0 \le j \le n-1$, or
(**E′**) $b_j = b_{j-1} + h_{j-1}(c_{j-1} + c_j)$, $1 \le j \le n$

From (**D**), we have

(**F**) $b_j = \frac{1}{h_j}(a_{j+1} - a_j) - \frac{h_j}{3}(2c_j + c_{j+1})$, $0 \le j \le n-1$, or
(**F′**) $b_{j-1} = \frac{1}{h_{j-1}}(a_j - a_{j-1}) - \frac{h_{j-1}}{3}(2c_{j-1} + c_j)$, $1 \le j \le n$.

Substitute (**F**) and (**F′**) into (**E′**), we have

(**G**) $h_{j-1}c_{j-1} + 2(h_{j-1} + h_j)c_j + h_j c_{j+1} = \frac{3}{h_j}(a_{j+1} - a_j) - \frac{3}{h_{j-1}}(a_j - a_{j-1})$, for $1 \le j \le n-1$.

Thus the problem is reduced to solving $A\mathbf{c} = \mathbf{h}$ with $(n-1)$ equations and $(n-1)$ unknown variables $\mathbf{c} = [c_1,\ c_2, \cdots,\ c_{n-1}]^t$ by using the boundary conditions $c_0 = \frac{1}{2}S''(x_0) = 0$ and $c_n = \frac{1}{2}S''(x_n) = 0$.

Once $\{c_j,\ 0 \le j \le n-1\}$ are solved, $\{d_j,\ 0 \le j \le n-1\}$ and $\{b_j,\ 0 \le j \le n-1\}$ could be easily solved by using (**C′**) and (**F′**), respectively.

$$\begin{bmatrix} 2(h_0+h_1) & h_1 & 0 & 0 & \cdots & 0 \\ h_1 & 2(h_1+h_2) & h_2 & 0 & \cdots & 0 \\ 0 & h_2 & \ddots & h_3 & \cdots & \vdots \\ \vdots & 0 & \vdots & \ddots & \cdots & 0 \\ \vdots & \vdots & 0 & h_{n-3} & \ddots & h_{n-2} \\ 0 & 0 & \cdots & \cdots & h_{n-2} & 2(h_{n-2}+h_{n-1}) \end{bmatrix} \times \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ \vdots \\ c_{n-1} \end{bmatrix} = \mathbf{h}$$

where

$$\mathbf{h} = \begin{bmatrix} \frac{3}{h_1}(a_2-a_1) - \frac{3}{h_0}(a_1-a_0) \\ \frac{3}{h_2}(a_3-a_2) - \frac{3}{h_1}(a_2-a_1) \\ \vdots \\ \vdots \\ \vdots \\ \frac{3}{h_{n-1}}(a_n-a_{n-1}) - \frac{3}{h_{n-2}}(a_{n-1}-a_{n-2}) \end{bmatrix}$$

Cubic Spline Interpolant for A Ruddy Duck

```
%
% Script File: cspline.m
% Cubic Spline Interpolation for a rubby duck of 21 points
%
n=21;
fin=fopen('duck.txt');
fgetL(fin);
X=fscanf(fin,'%f',n);
Y=fscanf(fin,'%f',n);
X0=0.9:0.4:13.3;
Y0=spline(X,Y,X0);
plot(X,Y,'b--o',X0,Y0,'r-'); axis([0.5 13.5, -1, 5]); grid
legend('Sample Points of A Duck','Cubic Spline Interpolant');
title('Cubic spline interpolant for a ruddy duck')
```

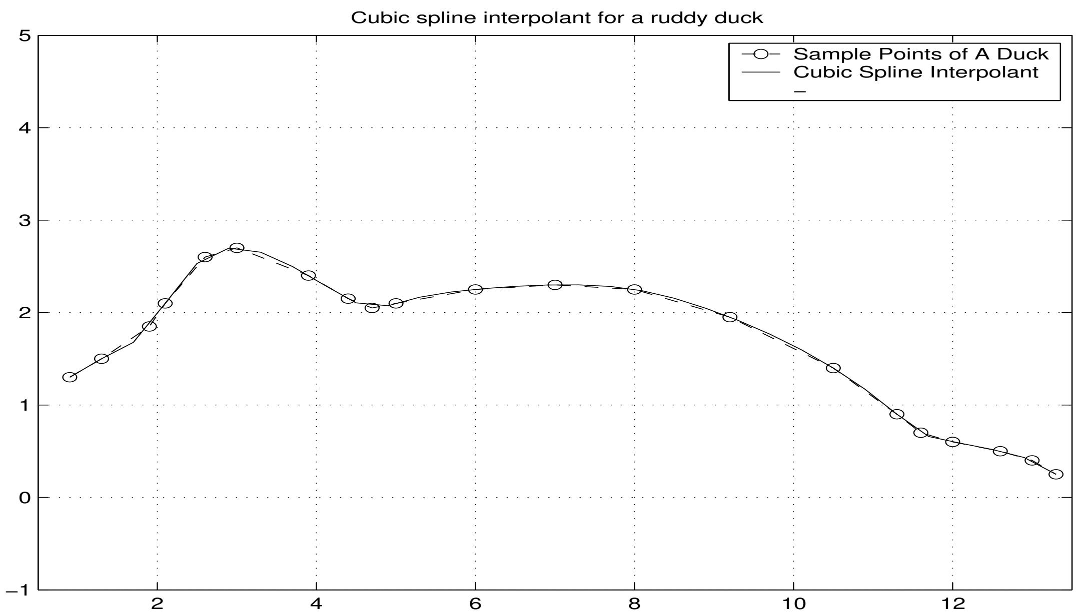

Figure 3: Cubic Spline Interpolant for A Ruddy Duck

Bezier Curves and B-splines

Bezier curves and B-splines are widely used in computer graphics and computer-aided design. These curves have good geometric property in that in changing one of the points we change only one portion of the fitted curve, a *local* effect. For cubic splines, changing only one point might have a *global* effect.

Bezier curves are named after the French engineer, Pierre Bezier of the Renault Automobile Company. He developed them in the early 1960's to fill a need for curves whose shape can be practically controlled by changing a few parameters.

The nth degree Bezier polynomial determined by $n+1$ points is given by

$$\mathbf{P}(u) \;=\; \sum_{i=0}^{n} C_i^m (1-u)^{n-i} u^i \mathbf{P}_i$$

Bezier cubics are commonly used. For $0 \le u \le 1$, denote

$$x(u) = (1-u)^3 x_0 + 3(1-u)^2 u x_1 + 3(1-u)u^2 x_2 + u^3 x_3$$

$$y(u) = (1-u)^3 y_0 + 3(1-u)^2 u y_1 + 3(1-u)u^2 y_2 + u^3 y_3$$

Then

$$\frac{dx}{du} = 3(x_1 - x_0), \quad \frac{dy}{du} = 3(y_1 - y_0) \;\; at \;\, u = 0.$$

$$\frac{dy}{dx} = \frac{y_1 - y_0}{x_1 - x_0} \;\; at \;\mathbf{P}_0, \quad \frac{dy}{dx} = \frac{y_2 - y_3}{x_2 - x_3} \;\; at \;\mathbf{P}_3$$

An Algorithm for drawing a Bezier curve

for $i = 0, 3n-1, 3$

    for $u \;=\; 0,\; 1,\; \Delta u$

        $x(u) = (1-u)^3 x_i + 3(1-u)^2 u x_{i+1} + 3(1-u)u^2 x_{i+2} + u^3 x_{i+3}$

        $y(u) = (1-u)^3 y_i + 3(1-u)^2 u y_{i+1} + 3(1-u)u^2 y_{i+2} + u^3 y_{i+3}$

        plot$(x(u), y(u))$

    endfor

endfor

B-splines

The B-splines (basis of splines) are like Bezier curves in that they do not ordinarily pass through the given data points. They can be of any degree, but cubic B-splines are commonly used.

Given the points $P_i(x_i,\ y_i)$, $i = 0, 1, \cdots, n$, a portion of a cubic B-spline for the interval $(P_i,\ P_{i+1})$, $i = 1, 2, \cdots, n-1$, is computed by

$$B_i(u) \ = \ \sum_{k=-1}^{2} b_k P_{i+k}$$

where

$$b_{-1} = \frac{(1-u)^3}{6}, \quad b_0 = \frac{u^3}{2} - u^2 + \frac{2}{3}, \quad b_1 = \frac{-u^3}{2} + \frac{u^2}{2} + \frac{u}{2} + \frac{1}{6}, \quad b_2 = \frac{u^3}{6}$$

u-cubics act as weighting factors on the coordinates of the four successive points to generate the curve, for example, at $u = 0$, the weights are $[\frac{1}{6}, \frac{2}{3}, \frac{1}{6}, 0]$; at $u = 1$, the weights are $[0, \frac{1}{6}, \frac{2}{3}, \frac{1}{6}]$.

An Algorithm for drawing a cubic B-spline

for $i = 1, n-2$

  for $u = 0, 1, \Delta u$

    $x = x_i(u)$

    $y = y_i(u)$

    plot(x,y)

  endfor

endfor

where

$$x_i(u) = \frac{(1-u)^3}{6} x_{i-1} + [\frac{u^3}{2} - u^2 + \frac{2}{3}] x_i + [\frac{-u^3}{2} + \frac{u^2}{2} + \frac{u}{2} + \frac{1}{6}] x_{i+1} + \frac{u^3}{6} x_{i+2}$$

$$y_i(u) = \frac{(1-u)^3}{6} y_{i-1} + [\frac{u^3}{2} - u^2 + \frac{2}{3}] y_i + [\frac{-u^3}{2} + \frac{u^2}{2} + \frac{u}{2} + \frac{1}{6}] y_{i+1} + \frac{u^3}{6} y_{i+2}$$

- Note that a B-spline does not necessarily pass through any point of $P_i's$.

Approximation Theory

Approximation theory deals with two types of problems.

- Given a data set, one seeks a function best fitted to this data set, for example, given $\{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}$, one seeks a line $y = mx + b$ which best fits this data set.

- Given an explicit function, one seeks a simpler function for representation, for example, use $1 + x + \frac{x^2}{2!} + \frac{x^3}{3!}$ to represent e^x.

• Orthogonal Functions

The set of functions $\{\phi_0, \phi_1, \cdots, \phi_n\}$ is said to be *orthogonal* for the interval $[a, b]$ with respect to the weight function w if

$$\int_a^b \phi_i(x)\phi_k(x)w(x)dx = \begin{cases} \alpha_k > 0 & if\ i = k \\ \\ 0 & if\ i \neq k \end{cases} \tag{2}$$

$\{\phi_0, \phi_1, \cdots, \phi_n\}$ is said to be *orthonormal* if, in addition, $\alpha_k = 1$ for $0 \leq k \leq n$.

♣ $\{1, \cos x, \sin x, \cdots, \cos kx, \sin kx, \cdots\}$ with respect to $w(x) \equiv 1$ is orthogonal for the interval $[0, 2\pi]$.

♣ $\{\frac{1}{\sqrt{2\pi}}, \frac{1}{\sqrt{\pi}}\cos x, \frac{1}{\sqrt{\pi}}\sin x, \cdots, \frac{1}{\sqrt{\pi}}\cos kx, \frac{1}{\sqrt{\pi}}\sin kx, \cdots\}$ with respect to $w(x) \equiv 1$ is orthonormal for the interval $[0, 2\pi]$.

♣ The set of Chebyshev polynomials $\{\cos(n\cos^{-1} x)\}_{n=0}^{\infty}$ is orthogonal with respect to $w(x) = \frac{1}{\sqrt{1-x^2}}$ for the interval $[-1, 1]$.

♣ The set of Chebyshev polynomials $\{\frac{1}{\sqrt{\pi}}, \frac{\sqrt{2}}{\sqrt{\pi}}[\cos(n\cos^{-1} x)]_{n=1}^{\infty}\}$ is orthonormal with respect to $w(x) = \frac{1}{\sqrt{1-x^2}}$ for the interval $[-1, 1]$.

♡ The set of Legendre polynomials $\{P_n(x) = \frac{1}{2^n n!} \cdot \frac{d^n(x^2-1)^n}{dx^n}\}$ is orthogonal with respect to $w(x) \equiv 1$ for the interval $[-1, 1]$. Note that

$$\int_{-1}^{1} P_m(x)P_n(x)dx = \begin{cases} \frac{2}{2n+1} & for\ m = n \\ \\ 0 & for\ m \neq n \end{cases} \tag{3}$$

Any high-order Legendre polynomial may be derived using the recursion formula

$$P_n(x) = \frac{2n-1}{n}xP_{n-1}(x) + \frac{n-1}{n}P_{n-2}(x) \tag{4}$$

Note that

$$P_0(x) = 1,\ P_1(x) = x,\ P_2(x) = \frac{1}{2}(3x^2 - 1),\ P_3(x) = \frac{1}{2}(5x^3 - 3x)$$