Hermite Interpolation Polynomial

Hermite interpolation polynomial allows us to find a polynomial that matches both function values and some of the derivative values at specified values on the independent variable.

Hermite polynomials were studied by the French Mathematician Charles Hermite (1822-1901), and are referred to as a "clamped cubic," where "clamped" refers to the slope at the endpoints being fixed.

The cubic Hermite polynomial p(x) has the interpolative properties

$$\begin{aligned} P(x_0) &= y_0, \\ P(x_1) &= y_1, \\ P'(x_0) &= d_0, \\ P'(x_1) &= d_1. \end{aligned}$$

Both the function values and their derivatives are known at the endpoints of the interval $[x_0, x_1]$ **.**

Theorem: (Cubic Hermite Polynomial) ***If f(x) is continuous on the interval*** $[x_0, x_1]$**, there exist a unique cubic polynomial**

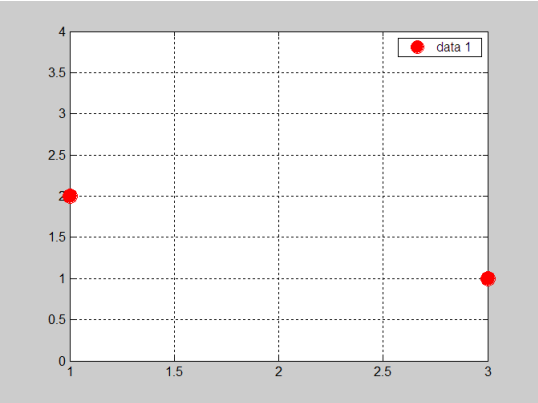
$$P(x) = ax^3 + bx^2 + cx + d$$

such that

$$\begin{aligned} P(x_0) &= y_0, \\ P(x_1) &= y_1, \\ P'(x_0) &= d_0, \\ P'(x_1) &= d_1. \end{aligned}$$

Example: Find the cubic Hermite polynomial that satisfies the given data

x	y=f(x)	y'
1	2	1
3	1	2



$$P(x) = ax^3 + bx^2 + cx + d$$

$$P(x_0) = y_0,$$
$$P(x_1) = y_1,$$
$$P'(x_0) = d_0,$$
$$P'(x_1) = d_1.$$

$$P(1)=2,$$
$$P(3)=1,$$
$$P'(1)=1,$$
$$P'(3)=2.$$

$$P'(x)=3ax^2+2bx+c$$

$$P(1)=a+b+c+d=2,$$
$$P(3)=27a+9b+3c+d=1,$$
$$P'(1)=3a+2b+c=1,$$
$$P'(3)=27a+6b+c=2.$$

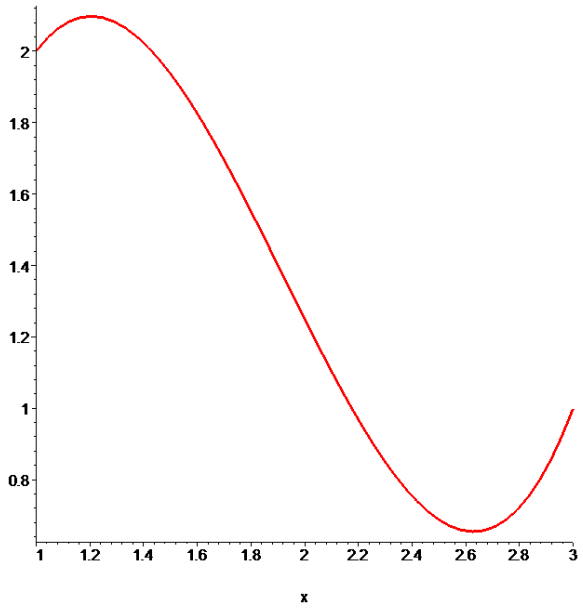
Linear Equation System

$$\begin{bmatrix}1 & 1 & 1 & 1\\ 27 & 9 & 3 & 1\\ 3 & 2 & 1 & 0\\ 27 & 6 & 1 & 0\end{bmatrix}\begin{bmatrix}a\\ b\\ c\\ d\end{bmatrix}=\begin{bmatrix}2\\ 1\\ 1\\ 2\end{bmatrix}$$

$$a=1, b=-\frac{23}{4}, c=\frac{19}{2}, d=-\frac{11}{4}$$

$$P(x)=ax^3+bx^2+cx+d$$

$$P(x)=x^3-\frac{23}{24}x^2+\frac{19}{2}x-\frac{11}{4}$$

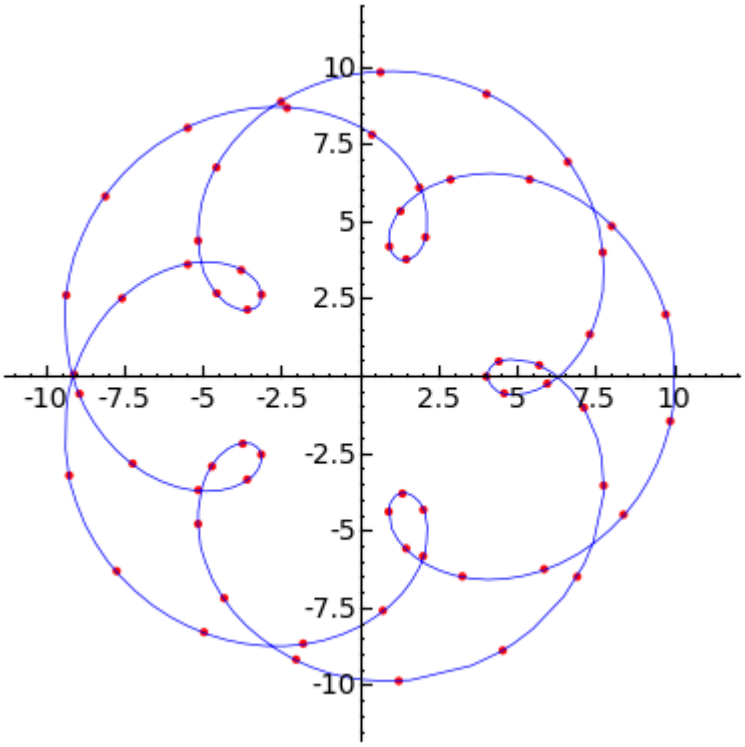


$$P(x) = x^3 - \frac{23}{24}x^2 + \frac{19}{2}x - \frac{11}{4}$$

MATLAB-Hermite Interpolation Polynomial

```
function a = hermite_coef(x, y, dy)
n = length(x);
a(1) = y(1);
for i = 1 : n
  xx(2*i-1) = x(i);
  yy(2*i-1) = y(i);
  xx(2*i) = x(i);
  yy(2*i) = y(i);
end
xx
yy
for k = 1 : n-1              % form first divided differences
  d(2*k-1, 1) = dy(k);
  d(2*k, 1) = (yy(2*k+1) - yy(2*k))/(xx(2*k+1) - xx(2*k));
end
d(2*n-1,1) = dy(n);
for j = 2 : 2*(n-1)
  for k = 1:2*n-j            % form jth divided differences
    d(k, j) = (d(k+1, j-1) - d(k, j-1))/(xx(k+j) - xx(k));
  end
end
d(1,2*n-1) = (d(2, 2*(n-1)) - d(1,2*(n-1)))/(xx(2*n) - xx(1));
d                            % display divided differences
for j = 2 : 2*n
  a(j) = d(1, j-1);
end
function p = hermite_eval(t, x, y, a)
n = length(x);
for i = 1 : n
  xx(2*i-1) = x(i);
  yy(2*i-1) = y(i);
  xx(2*i) = x(i);
  yy(2*i) = y(i);
end
for i = 1 : length(t)        % Evaluate Hermite polynomial at t
  ddd(1) = 1;                % Compute first term
  c(1) = a(1) ;
  for j = 2 : 2*n            % Compute jth term
    ddd(j) = (t(i) - xx(j-1)).*ddd(j-1);
    c(j) = a(j).*ddd(j);
  end
  p(i) = sum(c);
end
```

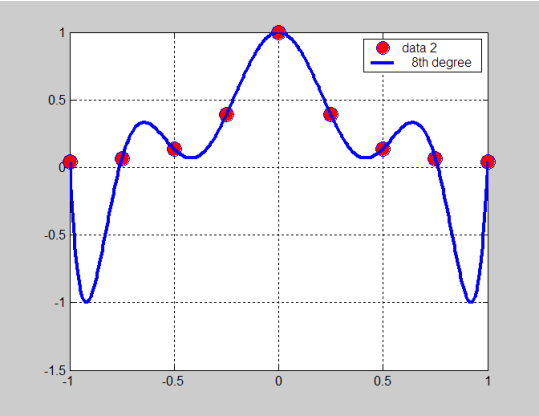
Spline Interpolation

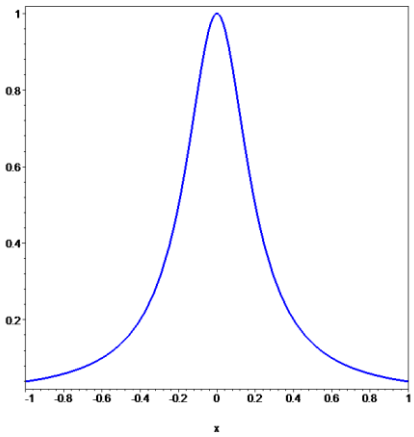


From Wikipedia

"In the computer science subfields of computer-aided design and computer graphics, the term spline more frequently refers to a piecewise polynomial (parametric) curve. Splines are popular curves in these subfields because of the simplicity of their construction, their ease and accuracy of evaluation, and their capacity to approximate complex shapes through curve fitting and interactive curve design."

To avoid the disadvantage of using a single polynomial to interpolate a large number of data points we can use piecewise polynomials.





From Wikipedia

“The term "Spline" is used to refer to a wide class of functions that are used in applications requiring data interpolation and/or smoothing. Splines may be used for interpolation and/or smoothing of either one-dimensional or multi-dimensional data. Spline functions for interpolation are normally determined as the minimizers of suitable measures of roughness (for example integral squared curvature) subject to the interpolation constraints. Smoothing Splines may be viewed as generalizations of interpolation splines where the functions are determined to minimize a weighted combination of the average squared approximation error over observed data and the roughness measure. For a number of meaningful definitions of the roughness measure, the spline functions are found to be finite dimensional in nature, which is the primary reason for their utility in computations and representation. For the rest of this section, we focus entirely on one-dimensional, polynomial splines and use the term "spline" in this restricted sense.”

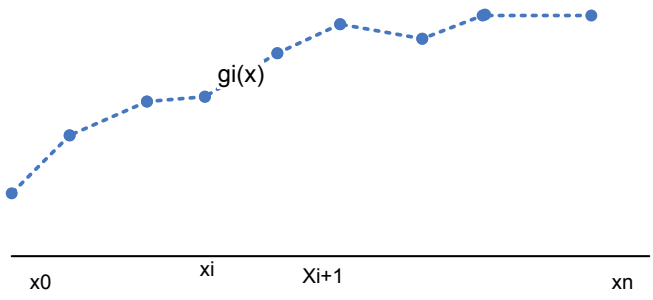
Piecewise Linear Interpolation

An alternative approach that can be used to obtain interpolation functions is to divide the interval into a collection of subintervals and construct a (generally) different approximating polynomial on each subinterval. Approximation by functions of this type is called piecewise polynomial approximation.

The simplest type of piecewise polynomial approximation is called piecewise linear interpolation (linear spline)

$$\{(x_0, f(x_0)),(x_1, f(x_1)),(x_2, f(x_2)),....,(x_n, f(x_n)\}$$

by a series of straight lines.



To illustrate the simplest form of piecewise polynomial interpolation, consider a set of four data points

$$(x_0, y_0), (x_1, y_1), (x_2, y_2), (x_3, y_3)$$
$$x_0 < x_1 < x_2 < x_3$$

These points define three subintervals on the x axis.

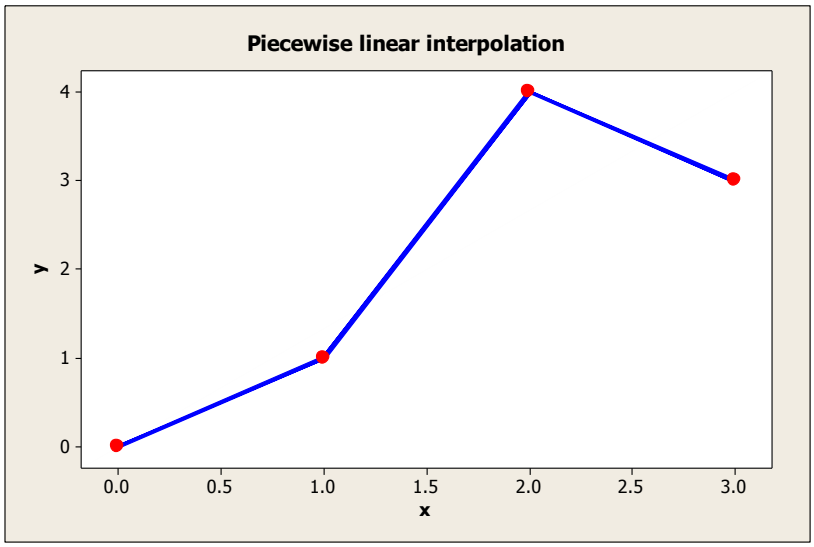
$$I_1 = [x_0, x_1], I_2 = [x_1, x_2], I_3 = [x_2, x_3]$$

If we use a straight line on each subinterval

$$P(x) = \begin{cases} \dfrac{(x - x_1)}{(x_0 - x_1)} y_0 + \dfrac{(x - x_0)}{(x_1 - x_0)} y_1, x_0 \le x \le x_1 \\ \dfrac{(x - x_2)}{(x_1 - x_2)} y_1 + \dfrac{(x - x_1)}{(x_2 - x_1)} y_2, x_1 \le x \le x_1 \\ \dfrac{(x - x_3)}{(x_2 - x_3)} y_2 + \dfrac{(x - x_2)}{(x_3 - x_2)} y_3, x_2 \le x \le x_3 \end{cases}$$

Using x= [0 1 2 3] and y= [0 1 4 3], we find the piecewise linear interpolation function illustrated in the following figure.

$$\begin{cases} x & x<1 \\ -2+3x & x<2 \\ 6-x & otherwi \end{cases}$$




```
> spline([0,1,2,3],[0,1,4,3],x,linear);
```

Piecewise Quadratic Interpolation

Using x= [0 1 2 3] and y= [0 1 4 3], we find the piecewise quadratic interpolation function.

```
> spline([0,1,2,3],[0,1,4,3],x,quadratic);
```

$$\begin{cases} \frac{26x^2}{35} & x \le \frac{1}{2} \\ \frac{9}{35} - \frac{36}{35}x + \frac{62}{35}x^2 & x \le \frac{3}{2} \\ -\frac{396}{35} + \frac{72}{5}x - \frac{118}{35}x^2 & x \le \frac{5}{2} \\ \frac{879}{35} - \frac{516}{35}x + \frac{86}{35}x^2 & otherwise \end{cases}$$

Example: x= [0 1 2 3] and y= [0 5 -1 0]

```
> Spline([[0,0],[1,5],[2,-1],[3,0]], x, degree=2,
endpoints='periodic');
```

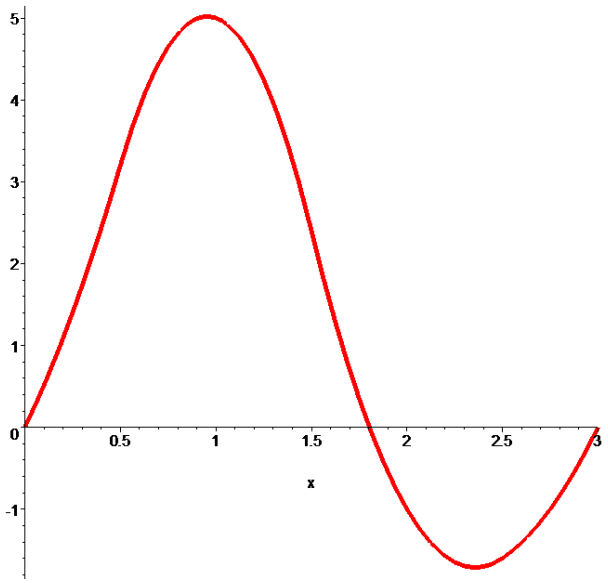
$$\begin{cases} \frac{24}{5}x + \frac{16}{5}x^2 & x \le \frac{1}{2} \\ -3 + \frac{84}{5}x - \frac{44}{5}x^2 & x \le \frac{3}{2} \\ \frac{147}{5} - \frac{132}{5}x + \frac{28}{5}x^2 & x \le \frac{5}{2} \\ \frac{72}{5} - \frac{72}{5}x + \frac{16}{5}x^2 & otherwise \end{cases}$$

Piecewise Function

```
> f:=piecewise(x<0.5,24/5*x+16/5*x^2,x<1.5,-3+84/5*x-
44/5*x^2,x<2.5,147/5-132/5*x+28/5*x^2,x<10,72/5-
72/5*x+16/5*x^2);
```

$$f := \begin{cases} \frac{24}{5}x + \frac{16}{5}x^2 & x < 0.5 \\ -3 + \frac{84}{5}x - \frac{44}{5}x^2 & x < 1.5 \\ \frac{147}{5} - \frac{132}{5}x + \frac{28}{5}x^2 & x < 2.5 \\ \frac{72}{5} - \frac{72}{5}x + \frac{16}{5}x^2 & x < 10 \end{cases}$$

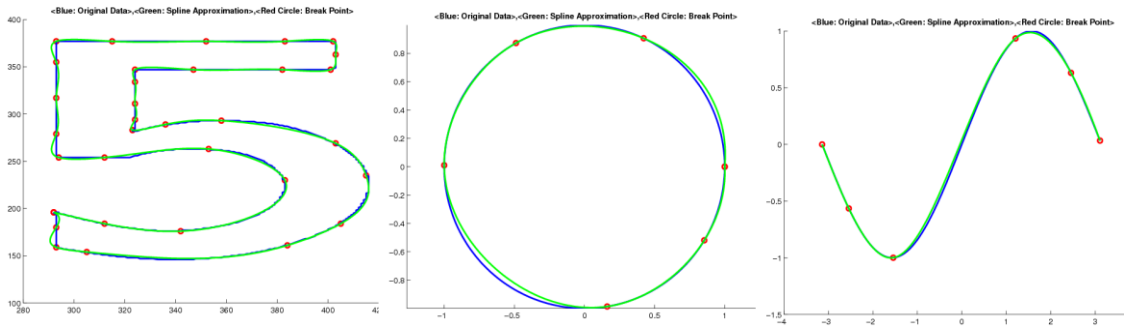
```
> plot(f,x=0..3);
```



- **The slopes are discontinuous where the segments join.**
- **The disadvantage of approaching an approximation problem using functions of this type is that at each of the endpoints of the subintervals, there is no assurance of differentiability, which, in a geometrical context, means that the interpolating function is not "smooth" at these points.**

Cubic Spline Interpolation

Piecewise Cubic Interpolation



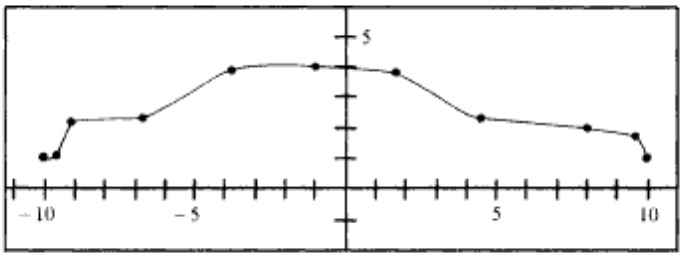


Figure 4a
Upper profile of a car

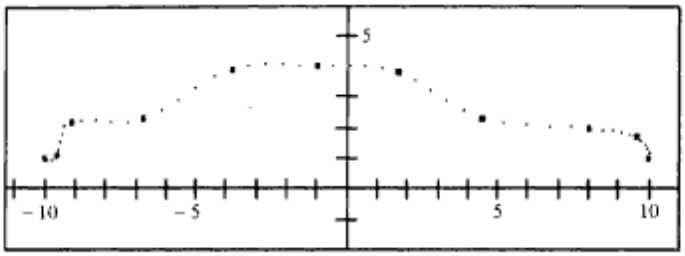


Figure 4b
Parametric fit to a car's profile

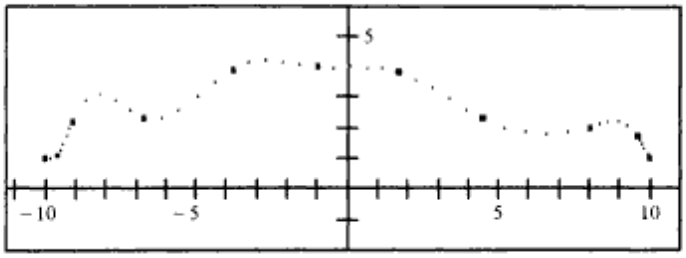


Figure 4c
Cubic spline fit to a car's profile

$P_{10}(9.6, 1.7)$, and $P_{11}(10, 1)$ were used to determine the parametric curve (Figure 4b) and the spline curve (Figure 4c).

Properties of the parametric curve. When a curve is pieced together from segments, its overall appearance depends on how well the segments fit together at the points P_k. Our curve C has two desirable properties that follow from the fact that the derivatives x', y' of the parametric functions for C are continuous even at the points P_k. Specifically,

$$x'(1; k-1, k) = x'(0; k, k+1) = .5(x_{k+1} - x_{k-1})$$

and

$$y'(1; k-1, k) = y'(0; k, k+1) = .5(y_{k+1} - y_{k-1}). \tag{9}$$

First, the curve is smooth (differentiable) at each point P_k. This follows from equations (9) because the derivative dy/dx at P_k is the same on the segment C_{k-1},

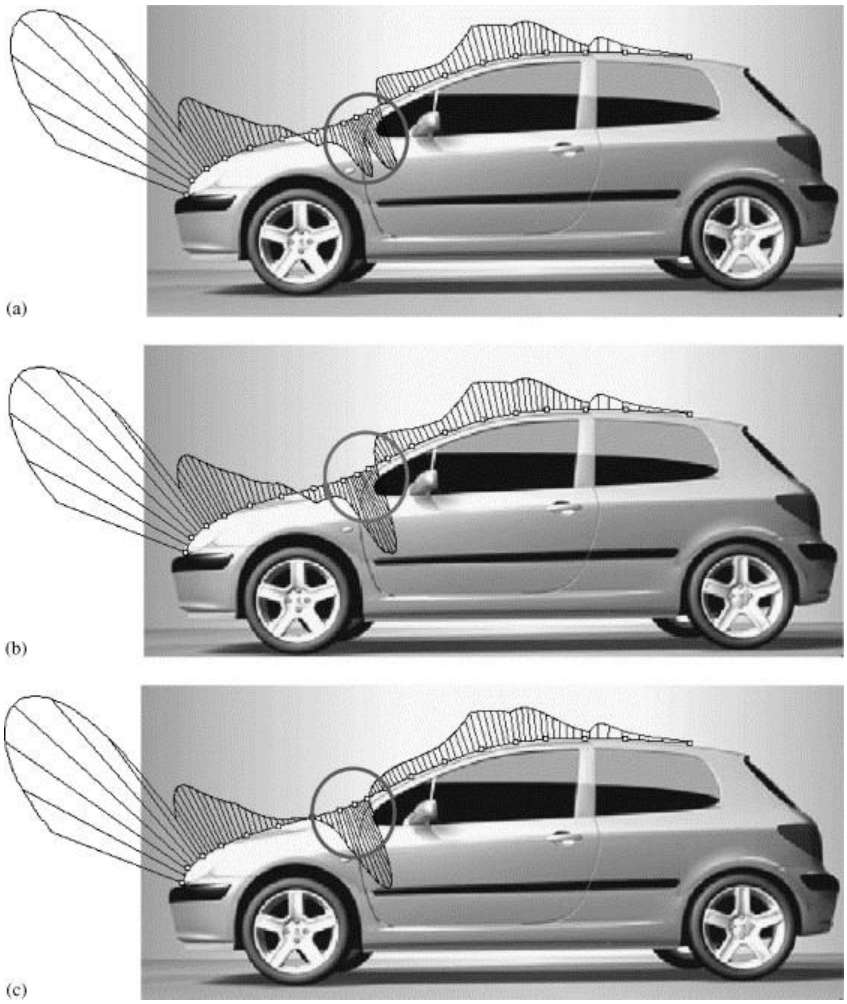


Fig. 7. (a) Cubic B-spline curve at level $j=4$ and its curvature distribution; (b) $\tilde{f}^4(u)$ with 9th CP thresholded with $k=0$; (c) $\tilde{f}^4(u)$ with 9th CP thresholded with $k=3$. (In (b) and (c) curvature distribution looks locally smoother.)

We will create a succession of cubic splines over successive intervals of the data. Each spline must join with its neighboring cubic polynomials at the knots where they join with the same slope and curvature.

We write the equation for a cubic polynomial $g_i(x)$, in the i^{th} interval, between points

$$(x_i, y_i), (x_{i+1}, y_{i+1}).$$

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

Thus, the cubic spline function we want is of the form

$$g(x) = g_i(x) \quad \text{on the interval } [x_i, x_{i+1}] \quad \text{for } i = 0,1,\ldots,n-1$$

and meets these conditions:

$$g_i(x_i) = y_i, \qquad i = 0,1,\ldots,n-1 \qquad \text{and} \quad g_{n-1}(x_n) = y_n,$$

$$g_i(x_{i+1}) = g_{i+1}(x_{i+1}), \qquad i = 0,1,\ldots,n-2;$$

$$g_i'(x_{i+1}) = g_{i+1}'(x_{i+1}), \qquad i = 0,1,\ldots,n-2;$$

$$g_i''(x_{i+1}) = g_{i+1}''(x_{i+1}), \qquad i = 0,1,\ldots,n-2.$$

Cubic Spline fits to each of the points, is

- **Continuous, and is**
- **Continuous in slope and**
- **Curvature throughout the region.**

If there is n+1 point, the number of intervals and the number of $g_i(x)$ are n. Thus there are four times n unknowns, which are the

$$\{a_i, b_i, c_i, d_i\}, i = 0,1,\ldots,n-1 .$$

Number of unknowns 4*n

The following equation

$$g_i(x_i) = y_i, \qquad i = 0,1,\ldots,n-1 \quad \text{and} \quad g_{n-1}(x_n) = y_n,$$

Immediately gives

$$d_i = y_i, i = 0,1,\ldots,n-1.$$

And the property

$$g_i(x_{i+1}) = g_{i+1}(x_{i+1}), \qquad i = 0,1,\ldots,n-2$$

gives

$$\begin{aligned} y_{i+1} &= g_{i+1}(x_{i+1}) = g_i(x_{i+1}) \\ &= a_i(x - x_{i+1})^3 + b_i(x - x_{i+1})^2 + c_i(x - x_{i+1}) + y_i \\ &= a_i h_i^3 + b_i h_i^2 + c_i h_i + y_i \quad \text{i} = 0,1,\ldots,\text{n -1}. \end{aligned}$$

$$h_i = (x_{i+1} - x_i).$$

To relate the slopes and curvatures of the joining splines, we differentiate the following equation

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

gives

$$g_i'(x) = 3a_i(x - x_i)^2 + 2b_i(x - x_i) + c_i,$$

$$g_i''(x) = 6a_i(x - x_i) + 2b_i \quad i = 0,1,\ldots,n-1.$$

Before we know

$$g_i''(x_{i+1}) = g_{i+1}''(x_{i+1}), \qquad i = 0,1,\ldots,n-2$$

If we let

$$S_i = g_i''(x_i) \quad \text{i} = 0,1,\ldots,\text{n}-1 \text{ and } S_n = g_{\text{n-1}}''(x_n)$$

We have

$$S_i = 6a_i(x_i - x_i) + 2b_i = 2b_i$$

$$S_{i+1} = 6a_i(x_{i+1} - x_i) + 2b_i = 6a_i h_i 2b_i$$

Hence we can write

$$b_i = \frac{S_i}{2},$$

$$a_i = \frac{S_{i+1} - S_i}{6h_i}.$$

$$d_i = y_i, i = 0,1,\ldots,n-1.$$

We substitute the relations a_i, b_i, d_i into

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

and then solve for c_i;

$$y_{i+1} = \left(\frac{S_{i+1} - S_i}{6h_i} \right) h_i^3 + \frac{S_i}{2} h_i^2 + c_i h_i + y_i$$

$$c_i = \frac{y_{i+1} - y_i}{h_i} - \frac{2h_i S_i + h_i S_{i+1}}{6}$$

SOLUTION:

$$b_i = \frac{S_i}{2},$$

$$a_i = \frac{S_{i+1} - S_i}{6h_i}$$

$$c_i = \frac{y_{i+1} - y_i}{h_i} - \frac{2h_i S_i + h_i S_{i+1}}{6}$$

$$d_i = y_i, i = 0,1,\ldots,n-1.$$

Solution of S_i

$$h_{i-1}S_{i-1} + (2h_{i-1} + 2h_i)S_i + h_i S_{i+1} = 6\left(\frac{y_{i+1} - y_i}{h_i} - \frac{y_i - y_{i-1}}{h_{i-1}}\right)$$
$$= 6\left(f[x_i, x_{i+1}] - f[x_{i-1}, x_i]\right)$$

Boundary Condition

(Free Boundary-Natural Spline)

$$g''(x_0) = g''(x_n) = 0$$

(Clamped Boundary-Clamped Cubic Spline)

$$g'(x_0) = f'(x_0) \quad \textbf{and} \quad g'(x_n) = f'(x_n)$$

NATURAL SPLINE

Take $S_0 = 0$ and $S_n = 0$. This makes the end cubics approach *linearity at their extremities*. This condition, called a natural spline. This technique is used very frequently.

$$h_{i-1}S_{i-1} + (2h_{i-1} + 2h_i)S_i + h_iS_{i+1} = 6\left(\frac{y_{i+1} - y_i}{h_i} - \frac{y_i - y_{i-1}}{h_{i-1}}\right)$$
$$= 6\left(f[x_i, x_{i+1}] - f[x_{i-1}, x_i]\right)$$

The matrix form of the equation

$$\begin{bmatrix} 2(h_0+h_1) & h_1 & & & \\ h_1 & 2(h_1+h_2) & h_2 & & \\ & h_2 & 2(h_2+h_3) & h_3 & \\ & & \ddots & & \\ & & & h_{n-2} & 2(h_{n-2}+h_{n-1}) \end{bmatrix} \begin{bmatrix} S_1 \\ S_2 \\ S_3 \\ \\ \\ S_{n-1} \end{bmatrix} =$$

$$6\begin{bmatrix} f[x_1,x_2] - f[x_0,x_1] \\ f[x_2,x_3] - f[x_1,x_2] \\ f[x_3,x_4] - f[x_2,x_3] \\ \\ \\ f[x_{n-1},x_n] - f[x_{n-2},x_{n-1}] \end{bmatrix}$$

Example:

The following Table lists values of a function.

i	x	f(x)
0	0.0	2.000
1	1.0	4.4366
2	1.5	6.7134
3	2.25	13.9130

Fit the data with a natural cubic spline curve, and evaluate the spline g(0.66) and g(1.75).

The true relation is $f(x) = 2e^x - x^2.$

We see that

$$h_0 = 1.0, h_1 = 0.5, h_2 = 0.75,$$

The divided differences that we can use to get the right-hand sides of our equations are

$$f[0,1] = 2.4366,$$
$$f[1,1.5] = 4.5536,$$
$$f[1.5,2.25] = 9.5995$$

For a natural cubic spline, we use $S_0 = 0$ and $S_3 = 0$.

$$\begin{bmatrix} 2(h_0+h_1) & h_1 \\ h_1 & 2(h_1+h_2) \end{bmatrix} \begin{bmatrix} S_1 \\ S_2 \end{bmatrix} = 6 \begin{bmatrix} f[x_1,x_2] - f[x_0,x_1] \\ f[x_2,x_3] - f[x_1,x_2] \end{bmatrix}$$

$$\begin{bmatrix} 3.0 & 0.5 \\ 0.5 & 2.5 \end{bmatrix} \begin{bmatrix} S_1 \\ S_2 \end{bmatrix} = \begin{bmatrix} 12.7020 \\ 30.2754 \end{bmatrix}$$

We obtain S_1=2.2920 and S_2=11.6518 (S_0=S_3=0)

Using these S's, we compute the coefficients of the individual cubic splines from the following equations.

$$b_i = \frac{S_i}{2},$$

$$a_i = \frac{S_{i+1} - S_i}{6h_i}$$

$$c_i = \frac{y_{i+1} - y_i}{h_i} - \frac{2h_i S_i + h_i S_{i+1}}{6}$$

$$d_i = y_i, i = 0,1,\ldots,n-1.$$

i interval $g_i(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i$

0 [0.0,1.0] $0.3820(x-0)^3 + 0(x-0)^2 + 2.0546(x-0) + 2.000$

1 [1.0,1.5] $3.1199(x-1)^3 + 1.146(x-1)^2 + 3.2005(x-1) + 4.4366$

2 [1.5,2.25] $-2.5893(x-1.5^3) + 5.8259(x-1.5)^2 + 6.6866(x-1.5) + 6.7134$

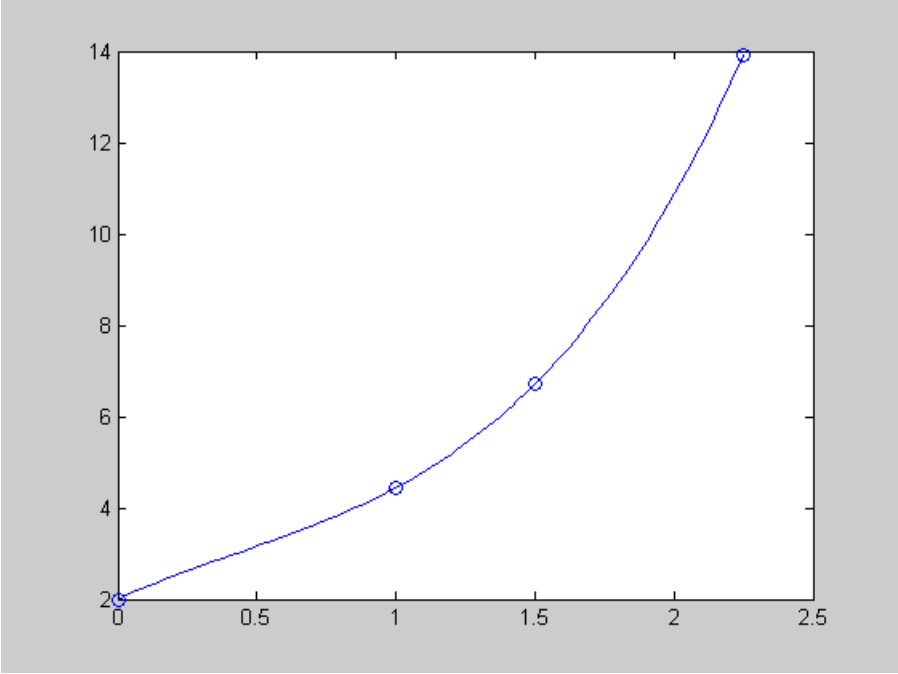
MAPLE SOLUTION

```
> spline([0,1,1.5,2.25],[2,4.4366,6.7134,13.9130],x,cubic);
```

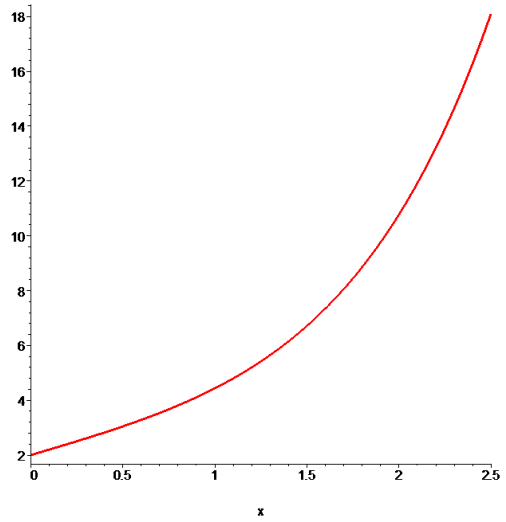
$$\begin{cases} 2. + 2.05459080500000014\,x + 0.382009195299999993\,x^3, & x < 1 \\ 1.235981609 + 3.20061839099999990\,x + 1.14602758620689650\,(x-1)^2 & \\ \quad + 3.11987126500000001\,(x-1)^3, & x < 1.5 \\ -3.316424140 + 6.68654942399999984\,x + 5.82583448275862104\,(x-1.5)^2 & \\ \quad - 2.58925976999999996\,(x-1.5)^3, & otherwise \end{cases}$$

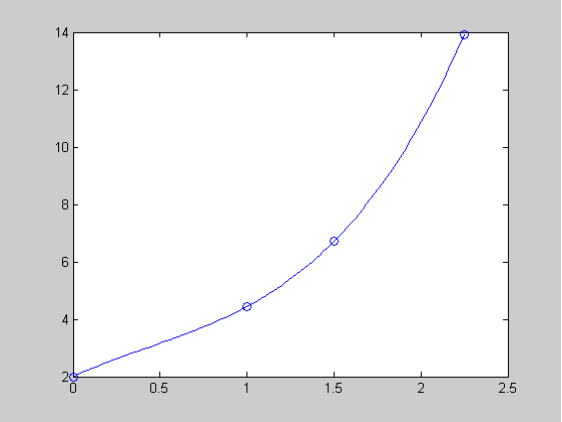
MATLAB SOLUTION

```
>>x=[0.0 1.0 1.5 2.25]
x =       0    1.0000    1.5000    2.2500
>> y=[2.000 4.4366 6.7134 13.9130]
y =    2.0000    4.4366    6.7134   13.9130
>> cs=csapi(x,y)
cs =
      form: 'pp'
    breaks: [0 1 1.5000 2.2500]
      coefs: [3x4 double]
    pieces: 3
      order: 4
       dim: 1
>> fnplt(cs);hold on; plot(x, y, 'o')
>>
```



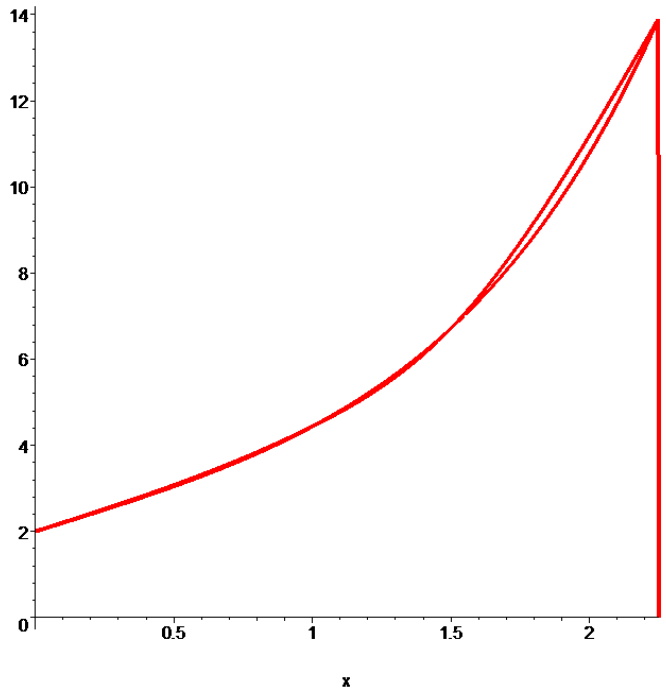
```
> f:=piecewise(x<1.0,0.3820*x^3+2.0546*x+2,x<1.5,3.1199*(x-1)^3+1.146*(x-1)^2+3.2005*(x-1)+4.4366,x<2.25,-2.5893*(x-1.5)^3+5.8259*(x-1.5)^2+6.6866*(x-1.5)+6.7134);
```





True Function **Cubic Spline**

$$f(x) = 2e^x - x^2.$$




```
> multiple(plot,[f,x=0.0..2.25],[2*exp(x)-x^2,x=0.0..2.25],thickness=4,colour=blue);
```

We use g_0 to find g(0.66)

$$g(0.66) = 0.3820(0.66 - 0)^3 + 0(0.66 - 0)^2 + 2.0546(0.66 - 0) + 2.000 = 3.4659$$

True value

$$f(0.66) = 2e^{0.66} - 0.66^2 = 3.4340$$

Using MATLAB

```
>> csapi(x,y, 0.66)
ans =
   3.5114
```

or

```
>> yi=interp1(x , y,0.66, 'spline')
yi =
   3.5114
```

Which are not identical our or MAPLE solutions. The reason is that MATLAB uses a different condition ("not a knot").

"Not a knot" condition:

Take S_0 as a linear extrapolation from S_1 and S_2, and S_n as a linear extrapolation from S_{n-1} and S_{n-2}.

We use g_2 to find g(1.75)

$$\begin{aligned} g(1.75) &= -2.5893(1.75-1.5)^3 + 5.8259(1.75-1.5)^2 \\ &\quad + 6.6866(1.75-1.5) + 6.7134 \\ &= 8.7087 \end{aligned}$$

True value

$$f(1.75) = 2e^{1.75} - 1.75^2 = 8.4467$$

Using MATLAB

```
>> csapi(x,y, 1.75)
ans =
   8.4993
```

OR

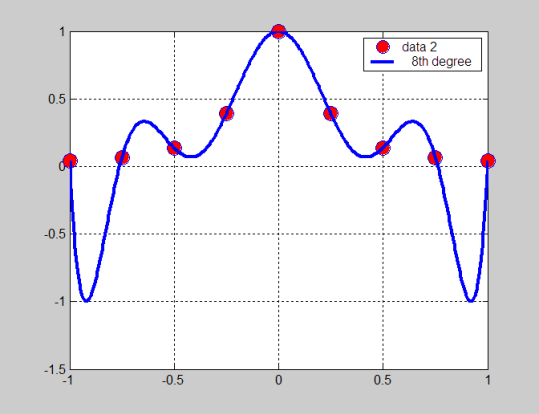
```
>> yi=interp1(x , y, 1.75, 'spline')
yi =
   8.4993
>>
```

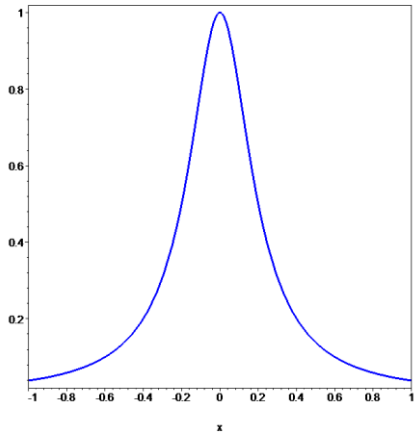
Example: RUNGE Function

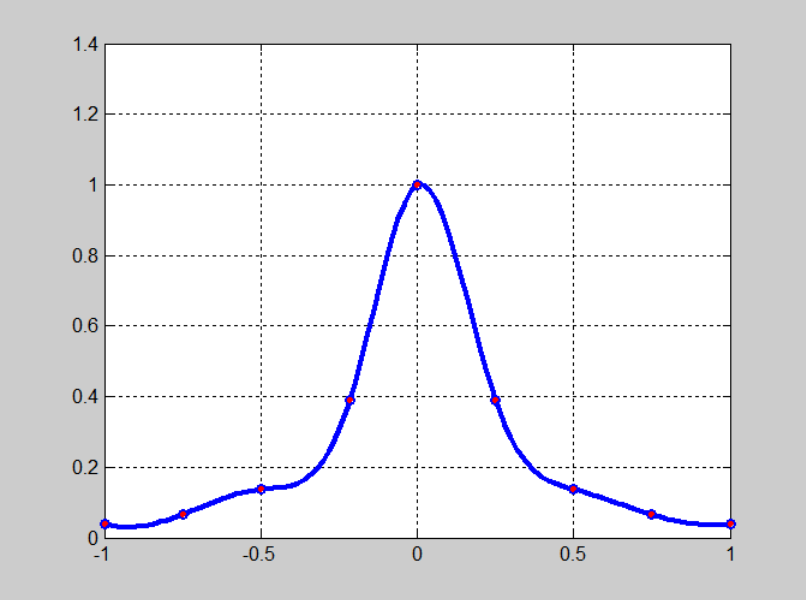
$$f(x) = \frac{1}{1+25x^2}$$

>>x=[-1 -0.75 -0.5 -0.215 0.0 0.25 0.50 0.75 1.00]

>> y=[0.0385 0.0664 0.138 0.3902 1.00 0.3902 0.138 0.0664 0.0385]







Cubic Spline Interpolation

```
cs=csapi(x,y)
cs =
      form: 'pp'
    breaks: [-1 -0.7500 -0.5000 -0.2150 0 0.2500 0.5000 0.7500 1]
     coefs: [8x4 double]
    pieces: 8
     order: 4
       dim: 1
>> fnplt(cs);hold on; plot(x, y, 'o')
```

Exercise:

The following Table lists values of a function.

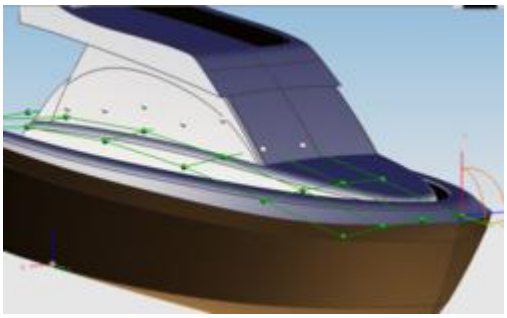
i	x	f(x)
0	0.0	0.0
1	1.0	0.5
2	2.0	2.0
3	3.00	1.5

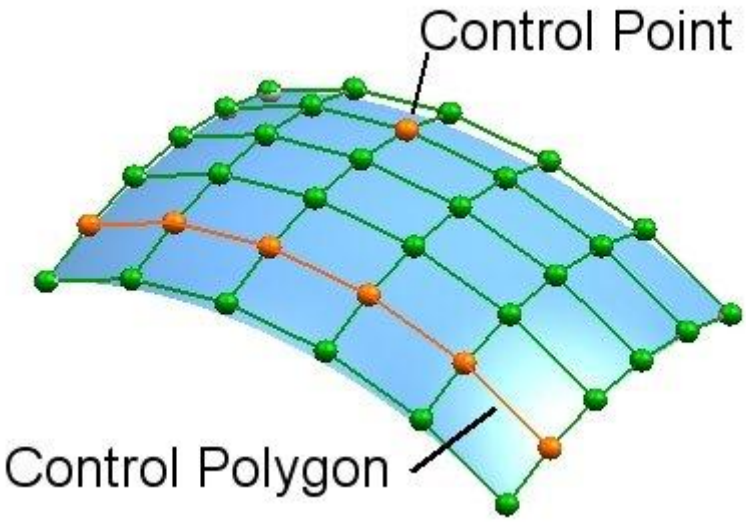
Fit the data with a natural Cubic Spline curve, and evaluate the Spline g(0.50) and g(2.5).

MATLAB-M File (Clamped Spline)

```
function S=csfit(X,Y,dx0,dxn)
%Input    - X is the 1xn abscissa vector
%         - Y is the 1xn ordinate vector
%         - dxo = S'(x0) first derivative boundary condition
%         - dxn = S'(xn) first derivative boundary condition
%Output   - S: rows of S are the coefficients for the cubic interpolants

N=length(X)-1;
H=diff(X);
D=diff(Y)./H;
A=H(2:N-1);
B=2*(H(1:N-1)+H(2:N));
C=H(2:N);
C=H(2:N);
U=6*diff(D);

%Clamped Spline endpoint constraints

B(1)=B(1)-H(1)/2;
U(1)=U(1)-3*(D(1)-dx0);
B(N-1)=B(N-1)-H(N)/2;
U(N-1)=U(N-1)-3*(dxn-D(N));
for k=2:N-1
  temp=A(k-1)/B(k-1);
  B(k)=B(k)-temp*C(k-1);
  U(k)=U(k)-temp*U(k-1);
end
M(N)=U(N-1)/B(N-1);
for k=N-2:-1:1
  M(k+1)=(U(k)-C(k)*M(k+2))/B(k);
end
%Clamped spline endpoint constraints

M(1)=3*(D(1)-dx0)/H(1)-M(2)/2;
M(N+1)=3*(dxn-D(N))/H(N)-M(N)/2;

for k=0:N-1
  S(k+1,1)=(M(k+2)-M(k+1))/(6*H(k+1));
  S(k+1,2)=M(k+1)/2;
  S(k+1,3)=D(k+1)-H(k+1)*(2*M(k+1)+M(k+2))/6;
  S(k+1,4)=Y(k+1);
end
```

Other Important Subjects





Bezier Curves and B-Spline Curves

- **Bezier curves and B-Splines are widely used in computer graphics and Computer-Aided Design (CAD).**
- **B-Splines are often used to numerically integrate and differentiate functions that are only through a set of data points.**

Bezier Surface

Wikipedia

Bézier surfaces were first described in 1972 by the French engineer Pierre Bézier who used them to design automobile bodies. Bézier surfaces can be of any degree, but bicubic Bézier surfaces generally provide enough degrees of freedom for most applications.

Cubic Bezier Surfaces

Bezier surface control points and polygon mesh: