Last updated: Yeh-Liang Hsu (2010-10-05).

Note: This is the course material for "ME550 Geometric modeling and computer graphics," Yuan Ze University. Part of this material is adapted from *CAD/CAM Theory and Practice*, by Ibrahim Zeid, McGraw-Hill, 1991. This material is be used strictly for teaching and learning of this course.

Curve representation

1. Wireframe models

There are three types of geometric models, wireframes, surfaces and solids. Typically, a wireframe model consists of a finite set of points (vertices), connected in pairs by straight lines (edges), or arcs, circles, conics, and curves, so that the three- dimensional form of a solid object can be visualized.

The major advantage of wireframe modeling is its simplicity to construct. It does not require as much computer time and memory as does surface or solid modeling. **Wireframe modeling is considered a natural extension of traditional methods of drafting.** Consequently, it does not require extensive training of users; nor does it demand the use of unusual terminology as surfaces and solids. Wireframe models form the basis for surface models. Most existing surface algorithms require wireframe entities to generate surfaces.

The disadvantages of wireframe models are manifolds. Primarily, **wireframe models are usually ambiguous representations of real objects and rely heavily on human interpretation.** A wireframe model of a box offers a typical example where the model may represent more than one object depending on which face(s) is assumed to exist. Models of complex designs having many edges become very confusing and perhaps even impossible to interpret. Moreover, as shown in Figure 1, it is often difficult to display objects with curve surfaces using wireframe.

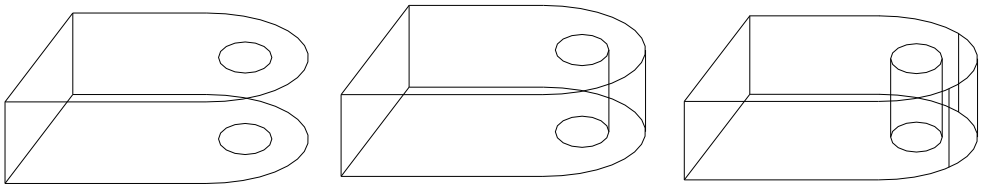


Figure 1. Displaying holes and curved ends in wireframe models

◇Assignment 1

Construct an object with several curve surfaces, the object in Figure 1 for example, using your CAD software. Display the object as a wireframe model. Change the views of the display. Do you find anything interesting on the curve surfaces? ◇

2. Parametric curves

2.1 Curve representation

Curves can be described mathematically by nonparametric or parametric equations. Nonparametric equations can be explicit or implicit. For a nonparametric curve, the coordinates *y* and *z* of a point on the curve are expressed as two separate functions of the third coordinate *x* as the independent variable [see Equation (1)]. This curve representation is known as the **nonparametric explicit form**. If the coordinates *x*, *y* and *z* are related together by two functions [see Equation (2)], a **nonparametric implicit form** results.

$$\mathbf{P} = \begin{bmatrix} x & y & z \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} x & f(x) & g(x) \end{bmatrix}^{\mathrm{T}} \tag{1}$$

$$\begin{aligned} F(x, y, z) &= 0 \\ G(x, y, z) &= 0 \end{aligned} \tag{2}$$

There are three problems with describing curves using nonparametric equations:

(1) **If the slope of a curve at a point is vertical or near vertical, its value becomes infinity or very large**, a difficult condition to deal with both computationally and programming-wise. Other ill-defined mathematical conditions may result.

(2) **Shapes of most engineering objects are intrinsically independent of any coordinate system.** What determines the shape of an object is the relationship between its data points themselves and not between these points and some arbitrary coordinate system.

(3) If the curve is to be displayed as a series of point or straight-line segments, the computations involved could be extensive.

Parametric representation allows closed and multiple-valued functions to be easily defined and replaces the use of slopes with that of tangent vectors, as will be introduced shortly.

In parametric form, each point on a curve is expressed as a function of a parameter *u*. The parametric equation for a three-dimensional curve in space takes the following vector form:

$$\mathbf{P}(u) = \begin{bmatrix} x & y & z \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} x(u) & y(u) & z(u) \end{bmatrix}^{\mathrm{T}}, \quad u_{\min} \le u \le u_{\max} \tag{3}$$

Equation (3) implies that the coordinates of a point on the curve are the components of its position vector. It is a one-to-one mapping from the parametric space (Euclidean space E^1 in *u* values) to the Cartesian space (E^3 in *x*, *y*, *z* values), as shown in Figure 2.

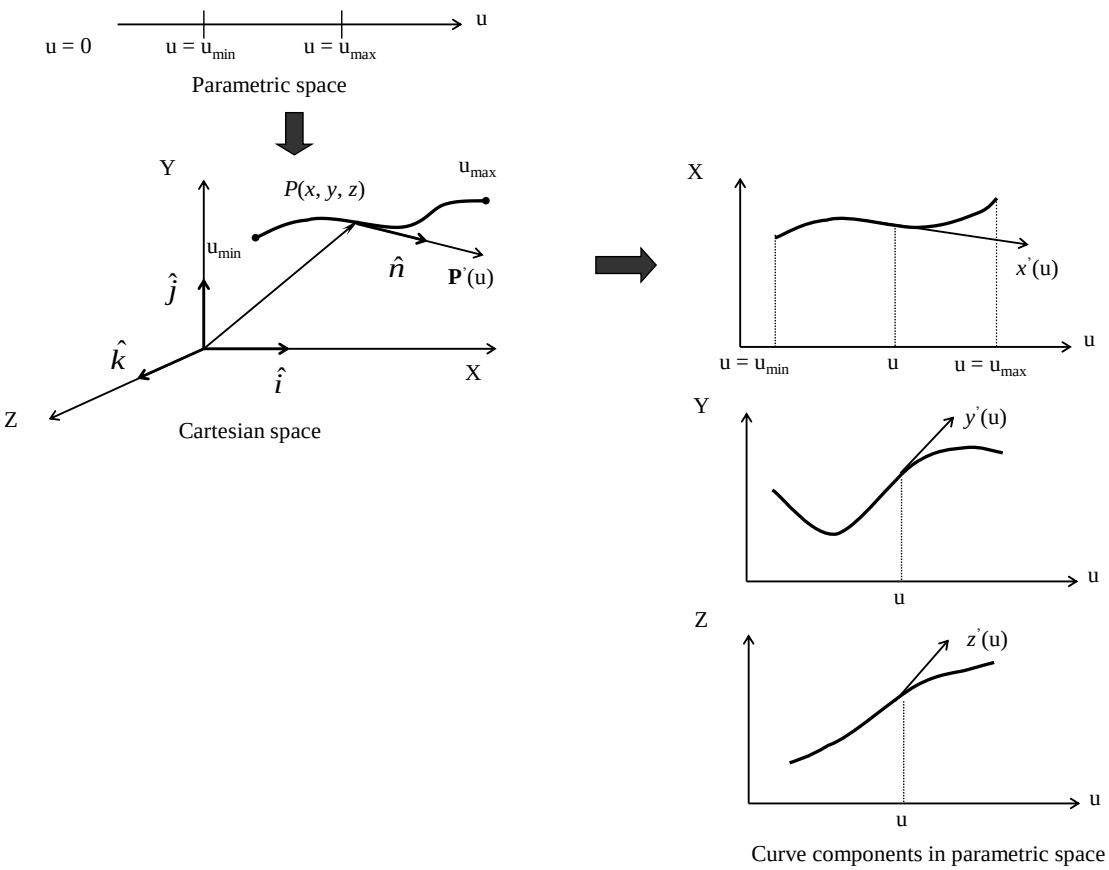


Figure 2. Parametric representation of a three-dimensional curve

The **tangent vector** is defined as vector

$$\mathbf{P}'(u) = \frac{d\mathbf{P}(u)}{du} \tag{4}$$

The components of the tangent vector in the parametric space as

$$\mathbf{P}'(u) = [x' \quad y' \quad z']^{\mathrm{T}} = [x'(u) \quad y'(u) \quad z'(u)]^{\mathrm{T}}, \quad u_{\min} \le u \le u_{\max} \tag{5}$$

where $x'(u)$, $y'(u)$, and $z'(u)$ are the first parametric derivatives (with respect to u) of the position vector components $x'(u)$, $y'(u)$, and $z'(u)$ respectively. The **slopes** of the curve are given by the ratios of the components of the tangent vector:

$$\begin{aligned} &\frac{dy}{dx} = \frac{dy/du}{dx/du} = \frac{y'}{x'} \\ &\frac{dz}{dy} = \frac{z'}{y'} \quad \text{and} \quad \frac{dx}{dz} = \frac{x'}{y'} \end{aligned} \tag{6}$$

2.2 Parametric representation of analytic curves

There are two categories of curves that can be represented parametrically: analytic curves and synthetic curves. **Analytic curves are defined as those that can be described by analytic equations** such as lines, circles, and conics. **Synthetic curves are the ones that are described by a set of data points (control points)** such as splines and Bezier curves.

Lines and circles are often expressed in analytic equations. They can also be expressed using parametric representation:

Lines

The parametric equation of the line becomes

$$\mathbf{P} = \mathbf{P}_1 + u(\mathbf{P}_2 - \mathbf{P}_1), \quad 0 \le u \le 1 \tag{7}$$

In scalar form, this equation can be written as

$$\left.\begin{aligned} x &= x_1 + u(x_2 - x_1) \\ y &= y_1 + u(y_2 - y_1) \\ z &= z_1 + u(z_2 - z_1) \end{aligned}\right\} \quad 0 \le u \le 1 \tag{8}$$

Note that Equation (7), a line is expressed as a synthetic curve.

◇Assignment 2

Give the coordinates of two end points of a line. Write a program in Matlab to draw the *x*, *y*, *z* components in parametric space (hint: use Equation (8)) and the isometric view of this line in Cartesian space (hint: use the transformation matrix discussed in the previous lecture, or use the Matlab built-in command for isometric view), as in Figure 2. Show your Matlab program too. ◇

Circles

Circles and circular arcs are among the most common entities used in wireframe modeling. Regardless of the user input information to create a circle, such information is always converted into a radius and center by the software.

The parametric equation of a circle can be written as

$$\left.\begin{aligned} x &= x_c + R\cos u \\ y &= y_c + R\sin u \\ z &= z_c \end{aligned}\right\},\quad 0 \le u \le 2\pi \tag{9}$$

For display purposes, Equation (9) can be used to generate points on the circle circumference by incrementing *u* from 0 to 360 degrees. These points are in turn connected with line segments to display the circles. However, this is an inefficient way due to computing the trigonometric functions in the equation for each point. A less computational method is to write Equation (9) in an incremental form.

$$\begin{aligned} x_n &= x_c + R\cos u \\ y_n &= y_c + R\sin u \\ x_{n+1} &= x_c + R\cos(u+\Delta u) \\ y_{n+1} &= y_c + R\sin(u+\Delta u) \\ z_{n+1} &= z_n \end{aligned} \tag{10}$$

Expanding the x_{n+1} and y_{n+1} equation gives

$$\begin{aligned} x_{n+1} &= x_c + (x_n - x_c)\cos\Delta u - (y_n - y_c)\sin\Delta u \\ y_{n+1} &= y_c + (y_n - y_c)\cos\Delta u + (x_n - x_c)\sin\Delta u \\ z_{n+1} &= z_n \end{aligned} \tag{11}$$

Thus, the circle can start from an arbitrary point and successive points with equal spacing can be calculated recursively. The increments $\cos\Delta u$ and $\sin\Delta u$ have to be calculated only once, which eliminates computation of trigonometric functions for each point.

◇Assignment 3

Derive Equation (9) to Equation (11) in more details. Decide a value of Δu and calculate $\cos\Delta u$ and $\sin\Delta u$. Write a Matlab program to generate the points of a circle using Equation (11), then connect the points to draw a circle. Try 3 different Δu and show the figures generated by your Matlab program. Show your Matlab program too. ◇

3. Synthetic curves

3.1 The need for synthetic curves

The need for synthetic curves in design arises on two occasions: when a curve is represented by a collection of measured data points and when an existing curve must change to meet new design requirements. Analytic curves are usually not sufficient to meet geometric design requirements of mechanical parts. Synthetic curves provide designers with greater flexibility and control of a curve shape by changing the positions of the control points. Products such as car bodies, ship hulls, airplane fuselage and wings, propeller blades, shoe insoles, and bottles are a few examples that require free-form, or synthetic, curves and surfaces. Considering that most data of objects are available in a discrete form, mainly key points, **the curve equation should be able to accept points and/or tangent values as input from the designer.**

Splines draw their name from the traditional draftsman's tool called "French curves or splines." In drafting terminology, a spline is a flexible strip used to produce a smooth curve through a designated set of points. Several small weights are distributed along the length of the strip to hold it in position on the drafting table. The term spline curve referred to a curve drawn in this manner.

We specify a spline curve by giving a set of coordinate positions, call control points, which indicate the general shape of the curve. These control points are then fitted with piecewise continuous parametric polynomial functions in one of the two ways. When polynomial sections are fitted so that the curve passes through each control point, the resulting curve is said **to interpolate the set of control points**. On the other hand, when the polynomials are fitted to the general control point path without necessarily

passing through any control point, the resulting curve is said **to approximate the set of control points**.

Interpolation curves are commonly used to digitize drawings or to specify animation paths. **Approximation curves are primarily used as design tools to structure object surfaces.**

Mathematically, synthetic curves represent a curve-fitting problem to construct a smooth curve that passes through given data points. Zero-order continuity (C^0) yields a position continuous curve. First (C^1)- and second (C^2)-order continuities imply slope and curvature continuous curves respectively. A C^1 curve is the minimum acceptable curve for engineering design. **A cubic polynomial is the minimum-order polynomial that can guarantee the generation of C0, C1 or C2 curves.**

Higher-order polynomials are not commonly used in CAD systems because they tend to oscillate about control points, are computationally inconvenient, and are uneconomical of storing curve and surface representations in the computer.

Also, the designer may prefer to control the shape of the curve locally instead of globally by changing the control point(s). **If changing a control point results in changing the curve locally in the vicinity of that point, local control of the curve is achieved; otherwise global control results.**

◇Assignment 4

Describe the synthetic curves supported by your CAD software. Draw a few 3d curves to show how you generate and modify these curves? Are they interpolation curves or approximation curves? ◇

3.2 Hermite Cubic Splines

The Hermite form of a cubic spline is determined by defining positions and tangent vectors at the data points, as shown in Figure 3.

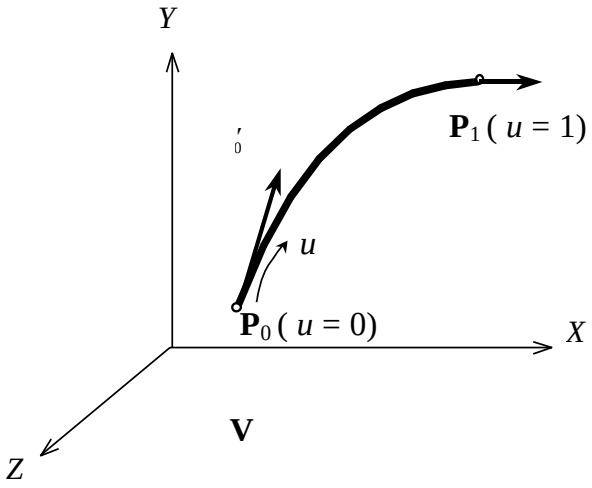


Figure 3. Hermite cubic spline curve

The parametric cubic spline curve (or cubic spline for short) connects two data (end) points and utilizes a cubic equation. Therefore, four conditions are required to determine the coefficients of the equation. The parametric equation of a cubic spline segment is given by

$$\mathbf{P}(u)=\sum_{i=0}^{3}\mathbf{C}_i u^i, \quad 0\le u\le 1 \tag{12}$$

where u is the parameter and $\mathbf{C}_i$ are the polynomial (also called algebraic) coefficients. In scalar form this equation is written as

$$\begin{aligned} x(u)&=C_{3x}u^3+C_{2x}u^2+C_{1x}u+C_{0x}\\ y(u)&=C_{3y}u^3+C_{2y}u^2+C_{1y}u+C_{0y}\\ z(u)&=C_{3z}u^3+C_{2z}u^2+C_{1z}u+C_{0z} \end{aligned} \tag{13}$$

In an expanded vector form, Equation (12) can be written as

$$\mathbf{P}(u)=\mathbf{C}_3u^3+\mathbf{C}_2u^2+\mathbf{C}_1u+\mathbf{C}_0 \tag{14}$$

This equation can also be written in a matrix form as

$$\mathbf{P}(u)=\mathbf{U}^{\mathrm{T}}\mathbf{C} \tag{15}$$

where $\mathbf{U} = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix}^T$ and $\mathbf{C} = \begin{bmatrix} \mathbf{C}_3 & \mathbf{C}_2 & \mathbf{C}_1 & \mathbf{C}_0 \end{bmatrix}^T$, and $\mathbf{C}$ is called the coefficients vector. The tangent vector is

$$\mathbf{P}'(u) = \sum_{i=0}^{3} \mathbf{C}_i i u^{i-1}, \quad 0 \le u \le 1 \tag{16}$$

Now the problem is how to relate the parametric equations to the designers' input, namely, the two end points and tangent vectors. Applying the boundary conditions ($\mathbf{P}_0$, $\mathbf{P}'_0$ at u=0 and $\mathbf{P}_1$, $\mathbf{P}'_1$ at u=1), Equations (12) and (16) give

$$\begin{aligned} \mathbf{P}_0 &= \mathbf{C}_0 \\ \mathbf{P}'_0 &= \mathbf{C}_1 \\ \mathbf{P}_1 &= \mathbf{C}_3 + \mathbf{C}_2 + \mathbf{C}_1 + \mathbf{C}_0 \\ \mathbf{P}'_1 &= 3\mathbf{C}_3 + 2\mathbf{C}_2 + \mathbf{C}_1 \end{aligned} \tag{17}$$

$$\begin{aligned} \mathbf{C}_0 &= \mathbf{P}_0 \\ \mathbf{C}_1 &= \mathbf{P}'_0 \\ \mathbf{C}_2 &= 3(\mathbf{P}_1 - \mathbf{P}_0) - 2(\mathbf{P}'_0 - \mathbf{P}'_1) \\ \mathbf{C}_3 &= 2(\mathbf{P}_0 - \mathbf{P}_1) + \mathbf{P}'_0 + \mathbf{P}'_1 \end{aligned} \tag{18}$$

Substituting Equation (18) into Equation (14) and rearranging gives

$$\mathbf{P}(u) = (2u^3 - 3u^2 + 1)\mathbf{P}_0 + (-2u^3 + 3u^2)\mathbf{P}_1 + (u^3 - 2u^2 + u)\mathbf{P}'_0 + (u^3 - u^2)\mathbf{P}'_1 \quad 0 \le u \le 1 \tag{19}$$

$$\mathbf{P}(u) = \mathbf{U}^T [\mathbf{M}_H] \mathbf{V}, \quad 0 \le u \le 1 \tag{20}$$

where $[\mathbf{M}_H]$ is the **Hermite matrix** and $\mathbf{V}$ is the geometry (or boundary conditions) vector. Both are given by

$$[\mathbf{M}_H] = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \tag{21}$$

$$\mathbf{V} = \begin{bmatrix} \mathbf{P}_0 & \mathbf{P}_1 & \mathbf{P}'_0 & \mathbf{P}'_1 \end{bmatrix}^T \tag{22}$$

◇Assignment 5

Use the same two end points in Assignment 2 and assume the tangent vectors on the end points. Using Equation (19), write a Matlab program to calculate the 11 points when u=0, 0.1, 0.2, …, 0.9, 1.0. Connect the 11 points to form the Hermite curve, and show its isometric view. Give 2 different sets of tangent vectors (different in direction and length). Draw the isometric view of the Hermite curves again. Try to properly define the tangent vectors so that the curves are three-dimensional. Describe your observations. Show the Matlab programs too.◇

3.3 Blending Cubic Spline Segments

Equation (19) is for one cubic spline segment. It can be generalized for any two adjacent spline segments of a spline curve that are to fit a given number of data points. This introduces the problem of blending or joining cubic spline segments which can be stated as follows: given a set of n points $\mathbf{P}_0$, $\mathbf{P}_1$, …, $\mathbf{P}_{n-1}$ and the two end tangent vectors $\mathbf{P}'_0$ and $\mathbf{P}'_{n-1}$ (as shown in Figure 4), connect the points with a cubic spline curve.

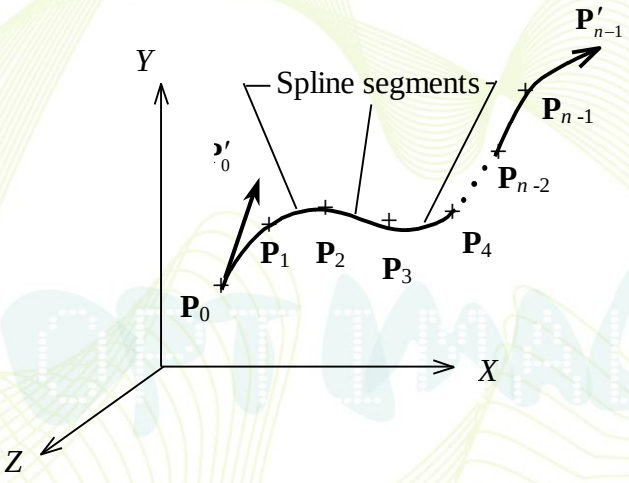


Figure 4. Hermite cubic spline curve

Let's start from connecting two curves. To connect two Hermite spline curves to form a C^2 continuous curve, the second derivative at the end of the first curve must be equal to the second derivative at the beginning of the first curve. Thus we have

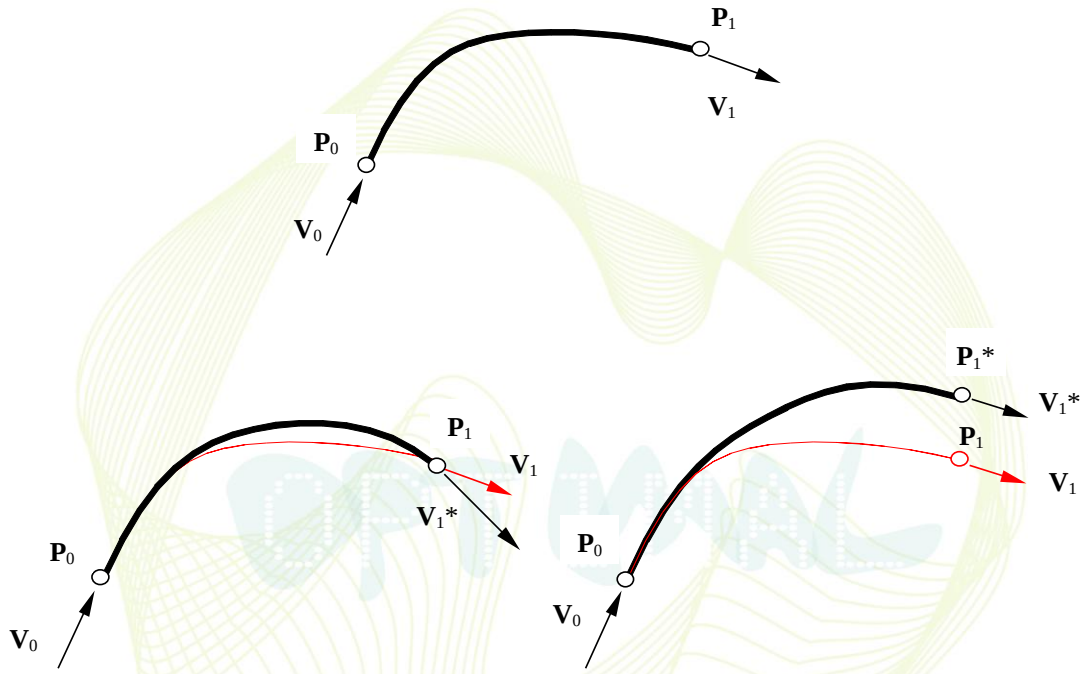
$$\mathbf{P}''(u_1 = 1) = \mathbf{P}''(u_2 = 0) \tag{23}$$

Using this relation, we can further derive the tangent vector at the end of the second curve, which is also the tangent vector at the beginning of the second curve:

$$\mathbf{P}'_1 = -\frac{1}{4}\left(3\mathbf{P}_0 + \mathbf{P}'_0 - 3\mathbf{P}_2 + \mathbf{P}'_2\right) \tag{24}$$

Using this information, we can construct a C^2 continuous curve that passes through the three given data points by blending two Hermite curves together. For N given data points, Equation (23) and (24) have to be used iteratively to obtain the tagent vector at the intermediate data points.

The use of the cubic splines in design applications is not very popular compared to Bezier or B-spline curves. **The control of the curve is not very obvious from the input data** due to its global control characteristics. As shown in Figure 5, **changing the position of a data point or an end slope changes the entire shape of the spline, which does not provide the intuitive feel required for design**. In addition, the order of the curve is always constant (cubic) regardless of the number of data points.



(a) Change of end slopes (b) Change of data points

Figure 5. Control of cubic spline curve

3.4 Lagrangian Interpolation Method

Often we have to display a curve interpolating through several points. Beside blending Hermite cubic spline segments, this can also be done by using the Lagrangian interpolation method. Equation (25) is the Lagrangian four-point form. It is easy to verify that at u=0, 1/3, 2/3, 1, the curve interpolates through $\mathbf{P}_0$, $\mathbf{P}_1$, $\mathbf{P}_2$, and $\mathbf{P}_3$.

$$\mathbf{P}(u)=\frac{1}{2}(1-u)(2-3u)(1-3u)\mathbf{P}_0 + \frac{9u}{2}(1-u)(2-3u)\mathbf{P}_1 + \frac{9u}{2}(1-u)(3u-1)\mathbf{P}_2 + \frac{u}{2}(2-3u)(1-3u)\mathbf{P}_3 \quad , \quad 0\le u\le 1 \tag{25}$$

◇Assignment 6

Verify that at u=0, 1/3, 2/3, 1, the curve in Equation (25) interpolates through $\mathbf{P}_0$, $\mathbf{P}_1$, $\mathbf{P}_2$, and $\mathbf{P}_3$. Assume the coordinates of the 4 control points. Write a Matlab program to calculate the 10 points when u=0, 1/9, 2/9, 3/9, …, 8/9, 1, using Equation (25), connect the 10 points to form the Lagrangian interpolation curve, and show its isometric view. Does the curve interpolate through the 4 control points? Show your Matlab program too.◇