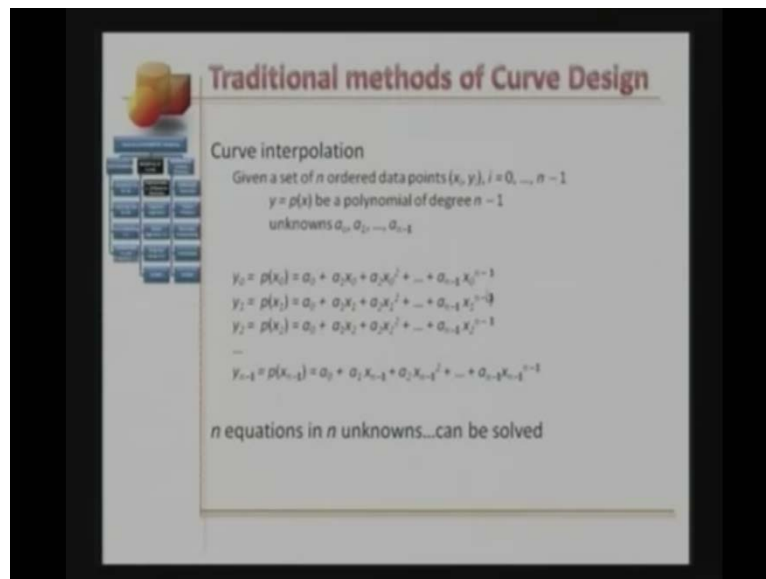


Computer Aided Engineering Design
Prof. Anupam Saxena
Department of Mechanical Engineering
Indian Institute of Technology, Kanpur

Lecture - 12

Welcome to lecture 12 of CAED. In this lecture we are going to be talking about design of curves, essentially we will touch upon 'Representation Techniques and Differential Geometry of Curves'. In the layout we are in the second column here on the design curves, why this curve designs important. Well, it forms the backbone of solid modeling; a network of curves as we know would represent surfaces; a set of surface patches forming a closed, simple and orient able surface represents a solid. This is a concept we have emphasized over and over again in previous lectures.

(Refer Slide Time: 01:06)



Traditional methods of Curve Design

Curve interpolation
Given a set of n ordered data points $(x_i, y_i), i = 0, \dots, n - 1$
 $y = p(x)$ be a polynomial of degree $n - 1$
unknowns a_0, a_1, \dots, a_{n-1}

$$y_0 = p(x_0) = a_0 + a_1 x_0 + a_2 x_0^2 + \dots + a_{n-1} x_0^{n-1}$$
$$y_1 = p(x_1) = a_0 + a_1 x_1 + a_2 x_1^2 + \dots + a_{n-1} x_1^{n-1}$$
$$y_2 = p(x_2) = a_0 + a_1 x_2 + a_2 x_2^2 + \dots + a_{n-1} x_2^{n-1}$$

...

$$y_{n-1} = p(x_{n-1}) = a_0 + a_1 x_{n-1} + a_2 x_{n-1}^2 + \dots + a_{n-1} x_{n-1}^{n-1}$$

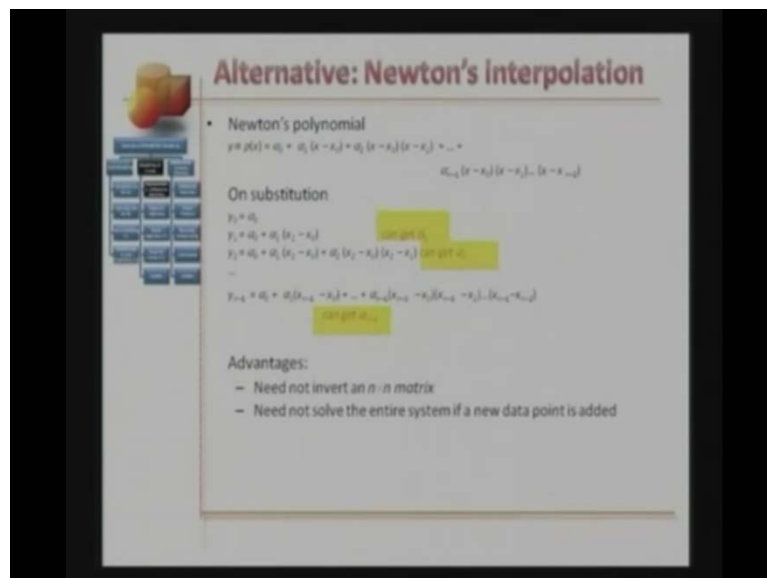
n equations in n unknowns...can be solved

Let us discuss in brief a few traditional methods of curve design. The first one is curve interpolation. Given a set of n ordered data points x_i, y_i, i going from 0 to n minus 1. We need to fix a polynomial y equals $p(x)$ which is a degree n minus 1. We all know from our previous classes and from our engineering first second third year classes how to do this. The polynomial will be of the type a_0 plus $a_1 x$ plus $a_2 x^2$ and so on so forth. The unknowns will be the coefficients a_0, a_1 until a_{n-1} .

As I said before this is how the polynomial is look like. $a_0 + a_1 x + a_2 x^2 + \dots + a_{n-1} x^{n-1}$. This is the polynomial when $x = 0$ is used in case of x ; clearly the left hand side will be the value y_0 . Likewise, if we substitute $x = x_1$, left hand side will be y_1 and the right hand side will be $a_0 + a_1 x_1 + a_2 x_1^2 + \dots + a_{n-1} x_1^{n-1}$.

We can keep on substituting the last equation will be y_{n-1} equals the value of the polynomial for $x = x_{n-1}$, which is $a_0 + a_1 x_{n-1} + a_2 x_{n-1}^2 + \dots + a_{n-1} x_{n-1}^{n-1}$. These are n equations n unknowns and we can solve them. We all know how to solve this. All we need to do is represent this set of equations in compact form and invert an $n \times n$ matrix. Notice that matrix inversion is an n^3 operation. If the number of equations that is the number of data points is large the version process will be slow.

(Refer Slide Time: 04:27)



An alternative is the Newton's interpolation technique. This avoids matrix inversion altogether the trick is to write the degree $n - 1$ polynomial slightly different. The polynomial $p(x)$ is expressed as $a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \dots + a_{n-1}(x - x_0)(x - x_1)\dots(x - x_{n-2})$. Notice that this term is a constant; this term is a constant; this is the term of

degree 1, this here is a term of degree 2, the next 1 will be a term of degree 3 and the last one will be a term of degree $n - 1$.

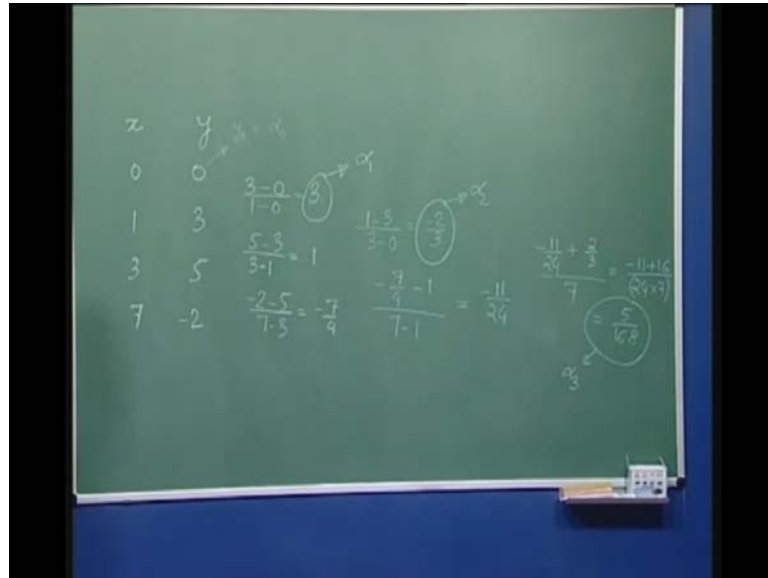
Notice also how these terms arranged for example, if I substitute $x = x_0$ all these terms become 0 leaving only α_0 . If I substitute $x = x_1$, the first term and the second term remain in series, while all the other terms they become zero. This allows to sequentially compute the unknowns α_0 , α_1 , α_2 and so on so forth. If we substitute $x = x_0$, we simply get $y_0 = \alpha_0$. As we said before all these terms will vanish if we substitute $x = x_1$ the third term fourth term and so on so forth will vanish leaving only the first and the second term right here. $\alpha_0 + \alpha_1(x_1 - x_0)$ this will be equal to y_1 . We already know α_0 from before using this equation we can compute α_1 .

For $x = x_2$, the first three terms remain the series while all the other terms become zero these are the corresponding terms $\alpha_0 + \alpha_1(x_2 - x_0) + \alpha_2(x_2 - x_0)(x_2 - x_1)$. We know α_0 and α_1 from the previous two equations using this equation you can compute α_2 . Notice again that these unknowns α_0 , α_1 , α_2 they are getting computed sequentially and not simultaneously as in the previous curve interpolation technique.

We can continue with this process and eventually get α_{n-1} . You will discuss this later but just repeat these coefficients α_0 , α_1 , α_2 , α_{n-1} . They are all called the divided differences. What are the advantages we do not need to invert an $n \times n$ matrix; once again we do not need to solve the entire system simultaneously. And if in case an additional data point (x_n, y_n) is added, we do not need to re-compute these coefficients, instead all we need to do is compute a new coefficient α_n .

Let us consider an example you need to construct a polynomial to interpolate through the data points $(0, 0)$, $(1, 3)$, $(3, 5)$ and $(7, -2)$ using the Newton's divided difference method. Let me show you an alternative very easy technique to compute coefficient. I will show this on the board; we compute the divided differences or the unknown coefficient column wise. The first thing we do is we arrange all the axis's and coordinates into columns like this.

(Refer Slide Time: 09:30)



The first point is 0 0, second point is 1 3, third point is 3 5, the fourth point is 7 minus 2. This value here is y 0 and you know that this is the first unknown alpha 0. To compute the first divide difference, what I do is I use these four numbers and compute the divide difference as 3 minus 0 which is this value minus this value over this minus this, 1 minus 0 which is 3.

Likewise I can compute the divide difference using these four numbers, how do I do that it is 5 minus 3 this minus this over 3 minus 1 which is equal to 1. Next, I use these four numbers here; the divide difference is minus 2 minus 5 over 7 minus 3, which is equal to minus 7 over 4. I will now fill the subsequent column next, what I do is, I will use these numbers and the corresponding in to compute the next values, this is how 1 minus 3 over. Now here, I am going to be skipping this entry and I am going to be considering this (()), 3 minus 0 which is equal to minus 2 over 3.

Likewise the divide difference here will be minus 7 over 4 minus 1 over. Now I am going to be considering this (()) 7 minus 1. This is equal to minus 11 over 24 and finally I will use these two numbers and I will use the entire (()) the first entry and the last entry. The corresponding divided difference will be minus 11 over 24 plus 2 over 3 over 7 minus 0. This turns out to be minus 11 plus 16 over 24 times 7, 24 7. This is equal to 5 over 168. Now where are my unknown alpha 0, alpha 1, alpha 2 and alpha 6. In this conversing triangle this is where the unknown coefficients are this one is alpha 1, this

one is alpha 2, and this one here is alpha 3, these unknowns up here on the top edge of the triangle.

(Refer Slide Time: 14:58)

Example: Newton's interpolation

Construct a polynomial to interpolate through the data points (0, 0), (1, 3), (3, 5) and (7, -2) using the Newton's divided difference

$$y = p(x) = \alpha_0 + \alpha_1(x - x_0) + \alpha_2(x - x_0)(x - x_1) + \dots + \alpha_3(x - x_0)(x - x_1)(x - x_2)$$

$$\alpha_0 = y_0 = 0$$

$$\alpha_1 = \frac{y_1 - \alpha_0}{x_1 - x_0} = 3$$

$$\alpha_2 = \frac{(y_2 - \alpha_0) - \alpha_1(x_2 - x_0)}{(x_2 - x_0)(x_2 - x_1)} = \frac{2}{3}$$

$$\alpha_3 = \frac{(y_3 - \alpha_0) - \alpha_1(x_3 - x_0) - \alpha_2(x_3 - x_0)(x_3 - x_1)}{(x_3 - x_0)(x_3 - x_1)(x_3 - x_2)} = \frac{5}{168}$$

(Refer Slide Time: 16:22)

- polynomial

$$y = 3x - \frac{2}{3}x(x-1) + \frac{5}{168}x(x-1)(x-3)$$

Moving data point (3, 5) to (3, 2) results in

$$\alpha_0 = y_0 = 0$$

$$\alpha_1 = \frac{y_1 - \alpha_0}{x_1 - x_0} = 3$$

$$\alpha_2 = \frac{(y_2 - \alpha_0) - \alpha_1(x_2 - x_0)}{(x_2 - x_0)(x_2 - x_1)} = -1.1667$$

$$\alpha_3 = \frac{(y_3 - \alpha_0) - \alpha_1(x_3 - x_0) - \alpha_2(x_3 - x_0)(x_3 - x_1)}{(x_3 - x_0)(x_3 - x_1)(x_3 - x_2)} = 0.1548$$

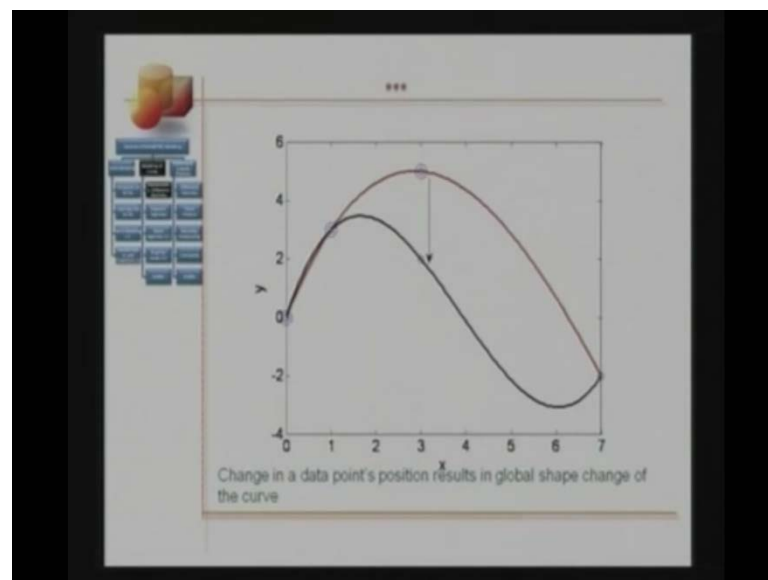
$$y_3 = 3x - 1.1667x(x-1) + 0.1548x(x-1)(x-3)$$

Coming back to the example one can also compute these unknowns the divide differences algebratly. We know alpha 0 is (()) y 0 which is 0 alpha one is y 1 minus alpha 0 over x 1 minus x 0 which is 3, alpha 2 is y 2 minus alpha 0, minus alpha 1 times x 2 minus x 0, over x 2 minus x 0 times x 2 minus x 1 , which is minus of 2 over 3 and alpha 3, we can work out the algebra, y 3 minus alpha 0 minus alpha 1, times x 3 minus x

0, minus alpha 2, times x 3, minus x 0, times x 3, minus x 1, over x 3, minus x 0 times x 3 minus x 1 times, x 3 minus x 2, which is 5 over 168.

We have seen these values before the corresponding polynomial can becomes y equals 3 times x minus 2 over 3 times x times x minus 1 plus 5 over 168 times x times x minus 1, times x minus 3. If I try to move the third data point 3 5 to a new location 3 2, this polynomial is going to change. The new coefficients are alpha 0 as 0 the same, alpha 1 as 3 the same, alpha 2 will be now minus 1.1667, and alpha 3 will be 0.1548 using the similar procedure. The new polynomial y n is 3 x minus 1.1667 times x times x minus 1 plus 0.1548 times x, times x minus 1 times x minus 3. Well, how do we see both these polynomials practically the curve.

(Refer Slide Time: 17:45)

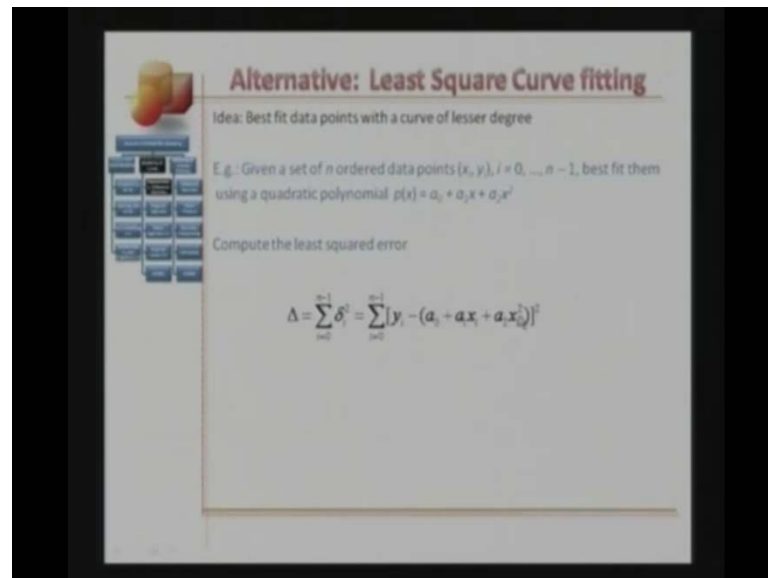


The color red is the old polynomial and if I move the third point to a new location the curve in black shows the new polynomial, what do we observe we see that change in a data point position results in global shape change of the curve. Remember we are interested in designing the curves and we would be more interested in knowing what happens when our design parameters in this example, the data points as we will solve them later, would change in position.

There are a few disadvantages of curve interpolation techniques. We would see more oscillations for higher number of data points; this is obviously a polynomial of degree n

will have at most n real roots. Graphically that polynomial is going to be intersecting the x axis at most n number of times. The more the n the more would be the oscillation the less the n or the less the degree of the curve the better for us. The second point global change in shape is observed if any data point is moved. In another words, curve interpolation methods do not offer local shape control to a design.

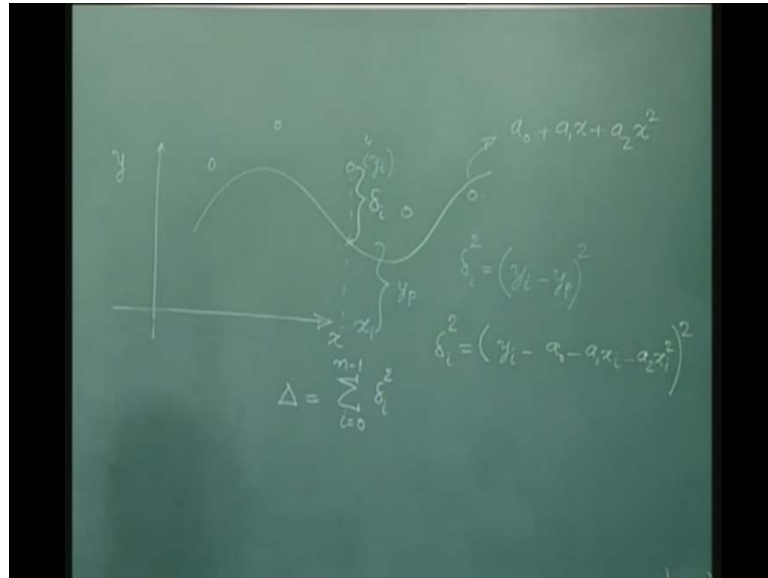
(Refer Slide Time: 20:03)



Well, we have an alternative to minimize oscillations, what we can do is use a lower degree curve. That would mean that we would forgo the ability for a curve to pass through data points, instead the curve will be very close to them the notion is to best fit data points with the curve of lesser degree. For example, given a set of n ordered data points $x_i y_i$ i going from 0 to n minus 1 you best fit them. Let us say using a quadratic polynomial $p(x) = a_0 + a_1x + a_2x^2$.

Well what we do is we would first compute the least squared error. The least squared error is given by capital delta which is equal to summation i going from 0 to n minus 1 of individual errors δ_i square which is equal to summation i going from 0 to n minus 1 y_i minus $a_0 + a_1x_i + a_2x_i^2$ the whole square. What would this equation mean graphically?

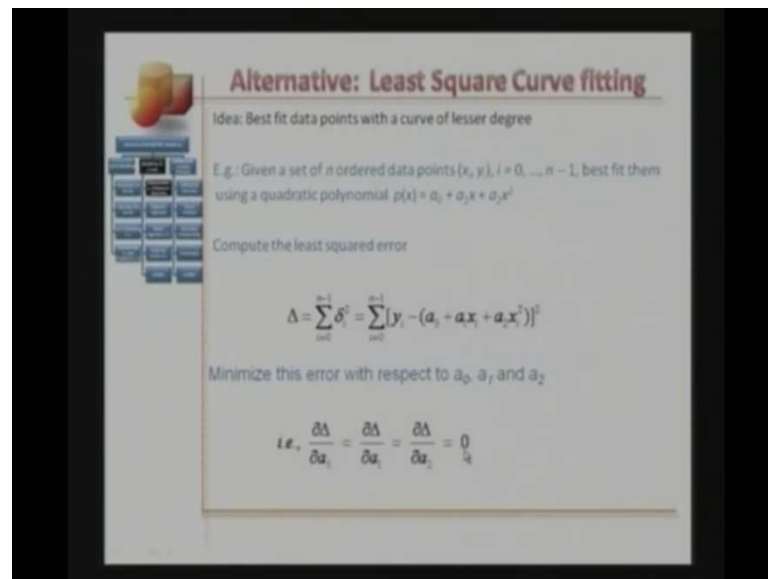
(Refer Slide Time: 21:26)



Let me explain on the board. Well, I start with the two dimensional coordinate axis x y; let us say we have a bunch of data points here. And we are trying to fit a quadratic polynomial, which would best depict the positions of these points. Let me take any point let us say this point i let me draw a vertical, this distance here is x i, this is the axis of this point and this distance the coordinate of this point is y i at x equals x i. The coordinate predicted by this quadratic polynomial is y p.

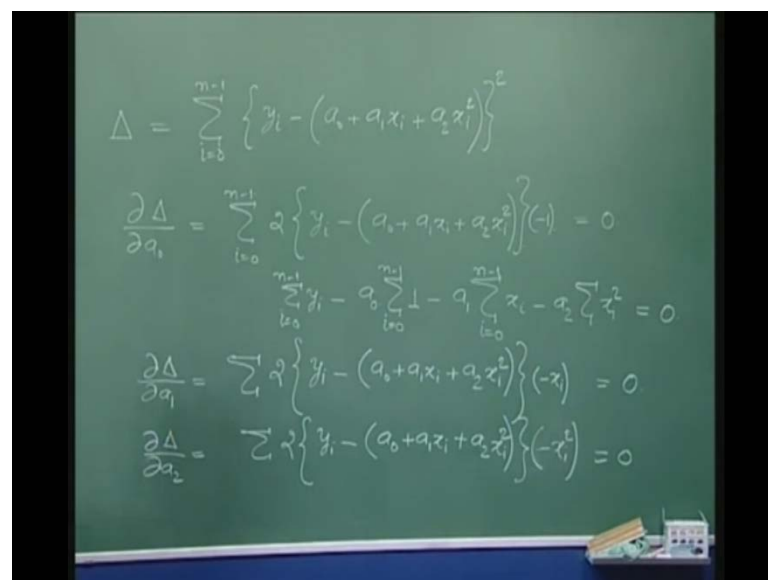
Naturally the error would correspond to this distance which is y i minus y p. One can either take the absolute value or one can take the square value; this is delta i square. Because that this is the quadratic column; so the predicted value y p for x equals x i will be given as, let me compute the error, this is y i minus y p will be a 0 minus a 1 x i minus a 2 x i square the whole square. This distance here again is delta i; naturally, the overall error capital delta will be the summation over all the points of small delta i.

(Refer Slide Time: 24:24)



Now how do we compute the unknowns a_0 , a_1 and a_2 ; well we minimize this error capital delta with respect these unknowns. How do we do that? We go back to basic calculus and we realize that if we treat these unknowns as independent variables, we can compute the slope partial delta over partial a_0 , partial delta over partial a_1 , partial delta over partial a_2 and make these slopes all go to 0.

(Refer Slide Time: 25:17)



Let me derive these equations for you here on board. So, partial delta over partial a_0 is given as summation i going from 0 to n minus 1, 2 times y_i minus a_0 plus a_1x_i plus $a_2x_i^2$

$2x_i^2 - 1$ and this is equal to 0, let me bring this summation inside and rewrite this equation as summation i going from 0 to $n-1$, $y_i - a_0 - a_1x_i - a_2x_i^2 = 0$. Let me drop the upper and lower limits here x_i^2 . This is equal to 0.

One can compute partial delta over partial a_1 in a similar fashion; as summation $2x_i y_i - a_0 - a_1x_i - a_2x_i^2 = 0$. Likewise partial capital delta over partial a_2 will be equal to summation $2x_i^2 y_i - a_0 - a_1x_i - a_2x_i^2 = 0$. Look like I have made a mistake here, I am computing partial derivative with respect to a_2 . So, this comes here will actually be minus of x_i^2 . So we have three equations one two and three and if you work out the second and third equation. By this simplifying then you would notice that you will have these three relations which are linear and the unknowns a_0 , a_1 and a_2 . So it is a three by three system we are talking about which is very easy to talk.

(Refer Slide Time: 29:40)

Example

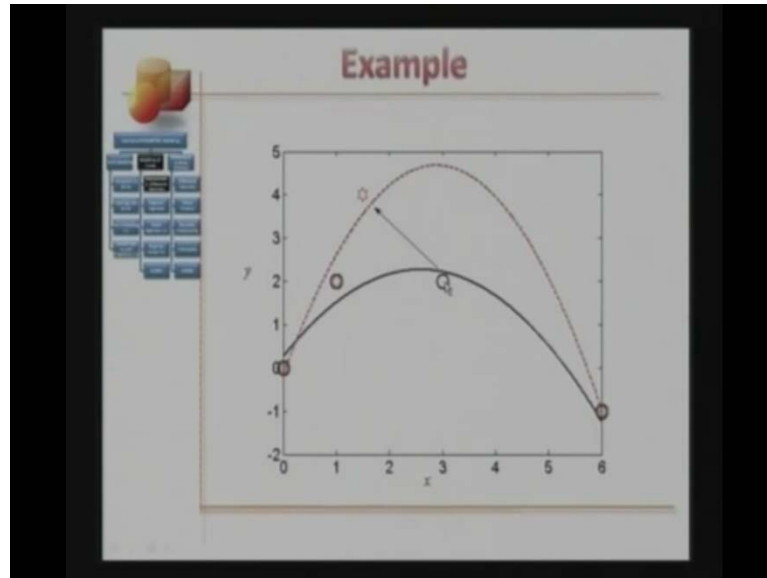
Construct a quadratic polynomial to best fit the data points (0, 0), (1, 2), (3, 2) and (6, -1):
 $y = 0.27 + 1.55x - 0.30x^2$

Construct a quadratic polynomial to best fit the data points (0, 0), (1, 2), (1.5, 4) and (6, -1):
 $y = -0.16 + 3.35x - 0.58x^2$

Well let us consider an example now; construct a quadratic polynomial to best fit the data points 0 0, 1 2, 3 2, 6 and minus 1. We have seen this now how to compute the unknowns the unknown coefficients of a quadratic polynomial; we will simply give you the results now. For these four data points the quadratic best fit polynomial is 0.27 plus

$1.55x - 0.30x^2$. If I move this point $(3, 2)$ to a new location say $(3, 4)$ and retain the locations of all these three points I get a new quadratic polynomial. The coefficients -0.16 , $3.35x$ and $-0.58x^2$.

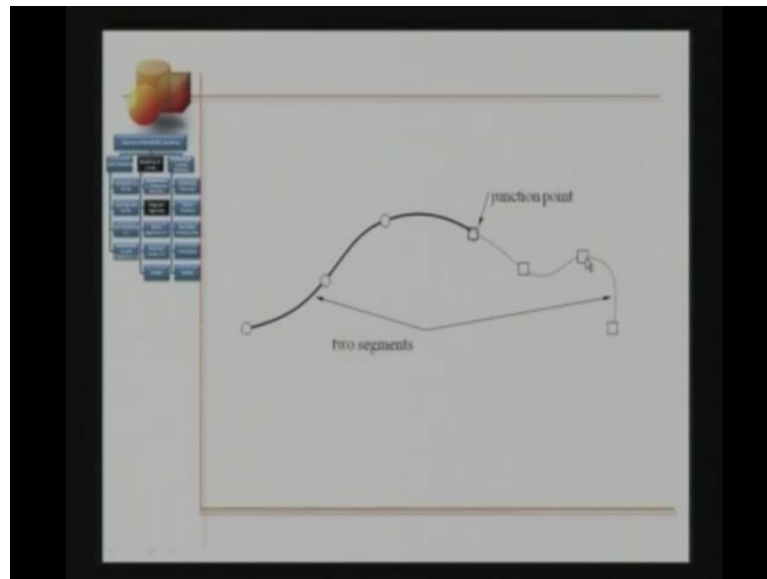
(Refer Slide Time: 30:48)



Let us see the two quadratic brackets. The black curve is the first quadratic equation and if I move this point to different location, I get this red curve as a new quadratic function. Once again we observe that changing the location of any data point affects the shape of the entire curve. Some features of curve fitting; well, the oscillations get reduced for higher number of data points which is expected then say using low order or low degree curves respect is given set of data points. But we still see global change in shape if any data point is moved. We still do not have local shape control on the curve what do we learn use of low degree curves to avoid unwanted fluctuations, and there is a possibility of the use of juxtaposed piecewise segments to facilitate local shape control of the curves.

Let me give you an example on the second point. Well, all we need to do is we need to consider the given set of data points or control points or design points, as we will call them later in different sub groups, each of smaller size. And then we can either interpolate or best fit points in individual sub groups using low degree segments. In this example we have taken four data points in first sub group and interpolated them using a cubic curve.

(Refer Slide Time: 32:41)



The next group is again a four data points again interpolated using a cubic segment. This point here is the junction point common to both segments and the two segments form four () composite curve. Well, let us say we try to relocate this point the shape of the cubic segment is expected to remain unchanged. It is a set of () conditions as the junction points that we will worry later.

(Refer Slide Time: 34:06)

Mathematical Representation

Explicit: $y = f(x)$
e.g. $y - y_1 = \frac{(y_2 - y_1)}{(x_2 - x_1)}(x - x_1)$

for $x_1 = x_2$, the line is vertical. The fact that y is non unique is not apparent

Implicit: $g(x, y) = 0$
e.g. line: $ax + by + c = 0$ and
circle: $x^2 + y^2 - r^2 = 0$

Finding their intersection requires that the equations be converted to explicit form

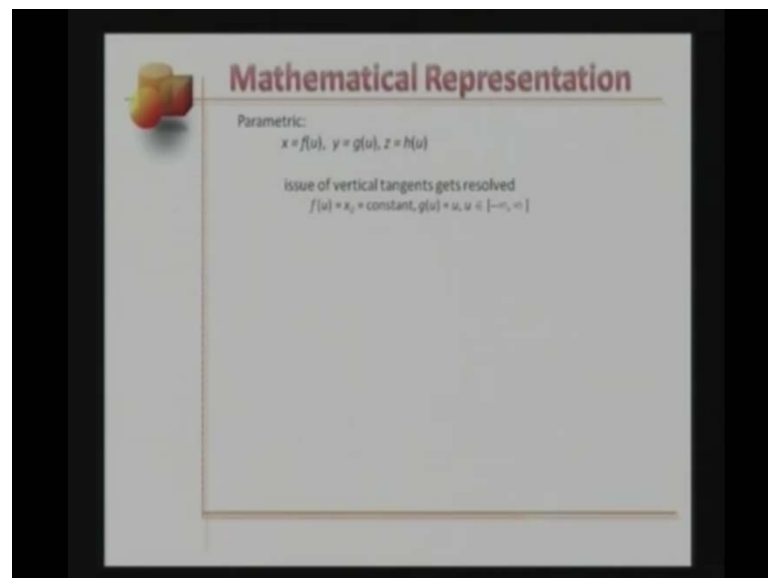
Further, two real (if existing) roots possible \Rightarrow extra processing

Let us now focus upon different mathematical representation techniques of curves. In general, we have explicit equations where a dependable variable y is explicitly expressed

in terms of an independent variable x . For example, we have the equation of a straight line from quadrant geometry $y - y_1 = \frac{y_2 - y_1}{x_2 - x_1} (x - x_1)$.

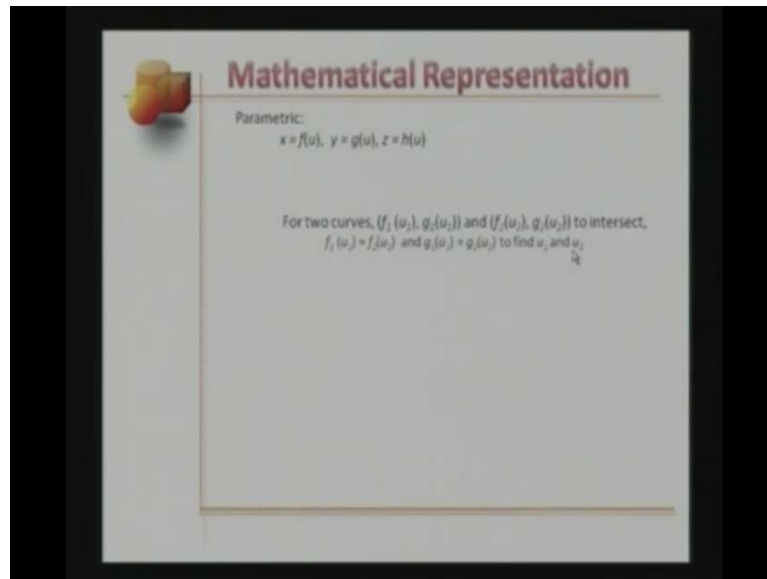
Well what happens, when $x_1 = x_2$ the line is vertical, the fact that y is not unique for this particular line is not very clear using this explicit representation. In fact this explicit representation cannot handle the condition $x_1 = x_2$. Another case implicit representation where independent and dependent variables are expressed together in a function; $g(x) + y = 0$. From this equation is not very clear which variable is dependent; for example, the equation for line $ax + by + c = 0$ and the equation of a circle $x^2 + y^2 - r^2 = 0$. If we find the intersection between this line and this circle the first thing you would need to do is we would need to convert these equations in the explicit form respectively and further if there are two real roots computing these roots will take extra mathematical processing

(Refer Slide Time: 36:21)



The third representation is the parametric one where the co-ordinates in the Euclidean space x, y and z are expressed each in terms of functions of an independent parameter u . For example x is f of u , y is g of u and z is h of u ; f, g and h are three functions of parameter u . The first advantage we get is that the issue of vertical tangents gets resolved; how? Well, one can simply say f of u is x_0 which is constant and g of u is equal to u where u goes from minus infinity to plus infinity.

(Refer Slide Time: 37:24)



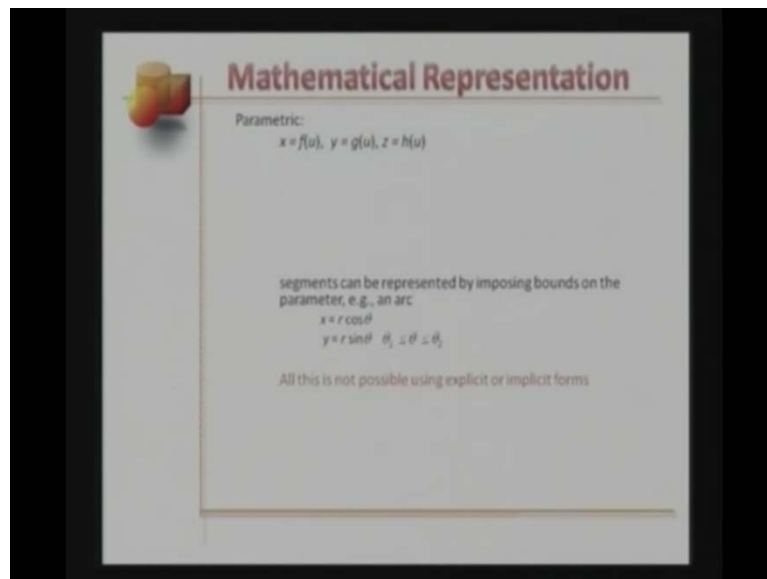
Mathematical Representation

Parametric:
 $x = f(u), y = g(u), z = h(u)$

For two curves, $(f_1(u_1), g_1(u_1))$ and $(f_2(u_2), g_2(u_2))$ to intersect,
 $f_1(u_1) = f_2(u_2)$ and $g_1(u_1) = g_2(u_2)$ to find u_1 and u_2

Further, if we have two curves let us say in two-dimensional space with functions f_1 and g_1 both expressed in terms of parameter u_1 and for the second curve we have functions, f_2 and g_2 both expressed in terms of a different parameter u_2 , we need to compute the intersection between these two curves. All we need to do say that the corresponding x is equal and the corresponding y are equal. In other words, f_1 of u_1 equals f_2 of u_2 and g_1 of u_1 equals g_2 of u_2 two conditions we give us values u_1 and u_2 .

(Refer Slide Time: 38:18)



Mathematical Representation

Parametric:
 $x = f(u), y = g(u), z = h(u)$

segments can be represented by imposing bounds on the parameter, e.g., an arc
 $x = r \cos \theta$
 $y = r \sin \theta \quad \theta_1 \leq \theta \leq \theta_2$

All this is not possible using explicit or implicit forms

The third advantage with parametric representation is that segments can be represented by imposing bounds on the parameter; for example, an arc of a circle parametrically circle is represented as $x = r \cos \theta$ and $y = r \sin \theta$ and if we impose bounds on the angle θ , θ_1 being the lower bound θ_2 being the upper bound. We would be able to represent a segment of a circle for now. This is not very much possible using either the explicit and implicit forms.

Using parametric representation the compact vector form of the parametric curve can be written as $\mathbf{r} = u$ note that \mathbf{r} is in bold represents the position vector of a point on the curve is equal to x of u and \mathbf{i} the instructor along the x direction plus y of u times \mathbf{j} the instructor along the y direction plus z of u times \mathbf{k} the instructor along the z or z direction.

The bottom line, we prefer to use parametric representation due to its advantages over the explicit and implicit representations number 1; number 2 we prefer to use low degree curves to minimize unwanted oscillations and number 3, instead of using a single curve, we choose a set of curves join them together or juxtapose them together at junction points, to form a composite curve which would represent a final design, that would either interpolate a best fit the given design point or data points.