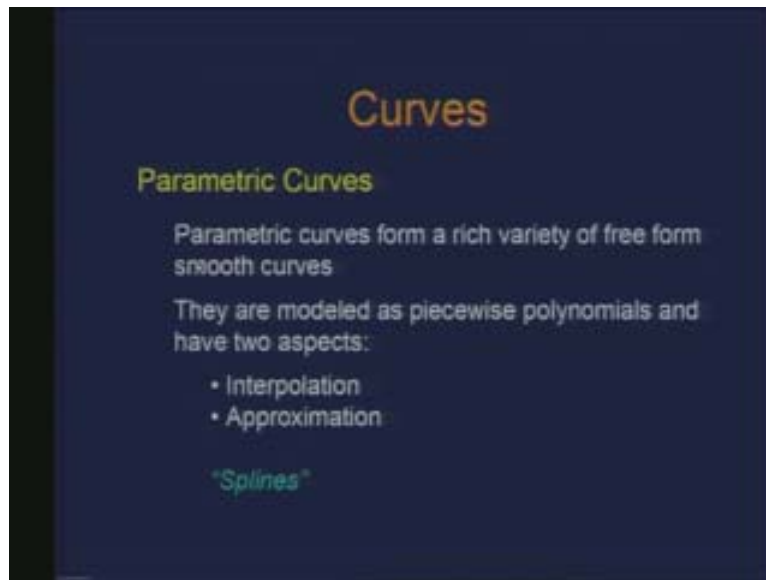


Introduction to Computer Graphics
Dr. Prem Kalra
Department of Computer Science and Engineering
Indian Institute of Technology, Delhi
Lecture # 12
Curves (Contd....)

We have been talking about parametric curves so just to review certain points about parametric curves what we have found is that parametric curves actually offer a rich variety of free form curves.

(Refer Slide Time: 01:12)

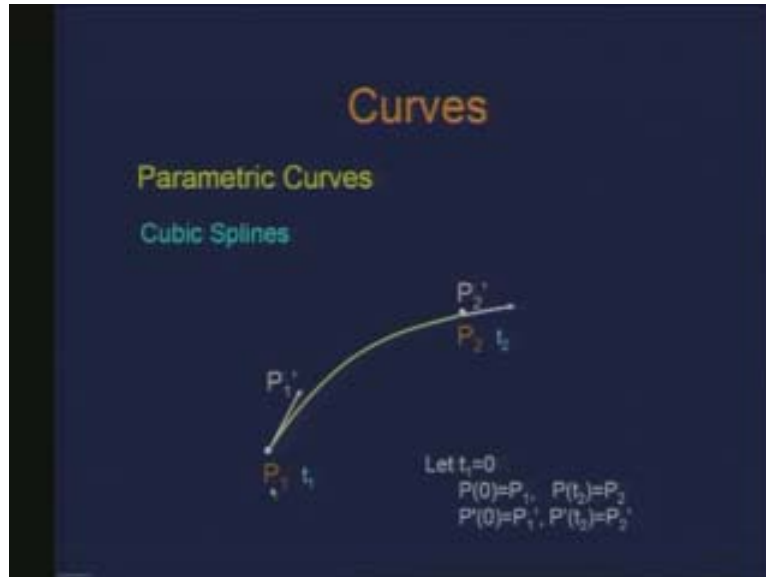


By free form we basically mean, a form which is not constrained by its representation. And these parametric curves basically offer the flexibility where we can freely change the form when we draw a curve. They are actually modeled as piecewise polynomials and we have two aspects; one is that they could be interpolating curves as we observed in the case of cubic splines the interpolating cubic splines or they could be approximation to a certain shape which is defined by certain input.

The idea is that give an approximation of the shape you want to obtain as a design and the curve which is drawn approximates that. So there is no exact notion of a particular shape but you have an approximation or the idea of the shape you want to have. So the splines which we looked at basically are the splines which are interpolating splines so the word spline which we observed was basically coming from a real life problem where the idea was to build these ships and big beams were laid out to get a certain shape which was desired for the ship design and those were called as splines.

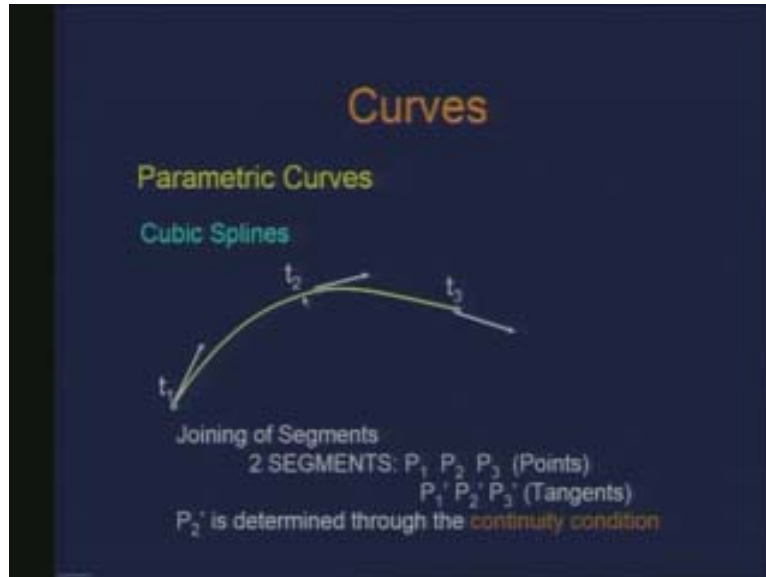
So, similar idea has been borrowed from there to construct these curves. So we looked at these cubic splines where we had the two end points like P_1 and P_2 and their tangent vectors which are basically the derivatives at that point P_1 and P_2 prime the derivative at the other end point P_2 .

(Refer Slide Time: 03:31)



So given these conditions as input also the running parameter t_1 and t_2 we construct a spline which passes through these points and satisfy the tangent vector conditions at the two end points and that is what we obtain cubic splines for. Now, if we extend the idea between the two points to any n number of points so there we are concerned of having these tangent vectors determined by certain constraint which is imposed on the curve rather than this being specified by the user.

(Refer Slide Time: 04:56)



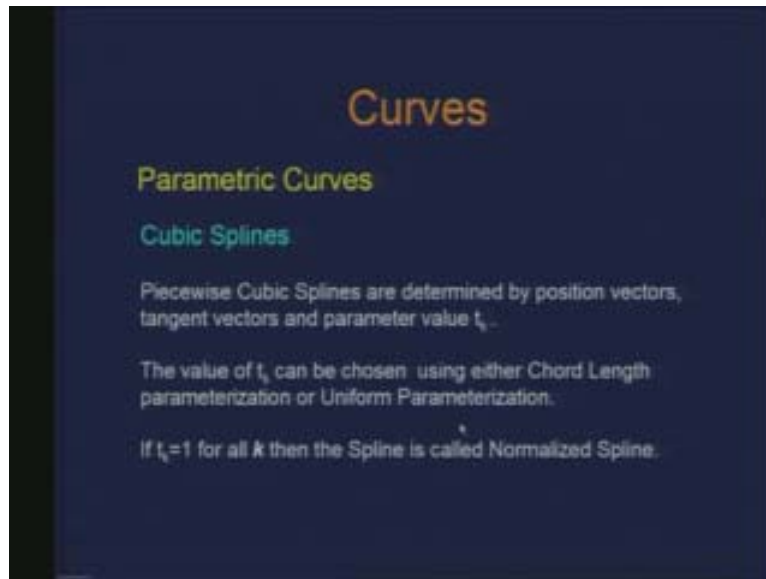
Therefore again only the two end tangent vectors are specified if desired and the intermediate tangent vectors are determined or computed using some continuity condition. And what we used was that the c_2 continuity condition in the case of a cubic spline where we say that the second derivative is the same at the joining point.

So at this point we say that the second derivative for the first curve and the second derivative for the second curve at this point is equal and from there we derive the tangent vector and this then can be extended to any set of points. So basically this sets up a system of equation and we solve for all the intermediate tangent vectors.

Let us look at what is the input to the specification of cubic splines and how we can change the spline in terms of its shape and what are the parameters. So we observe that the input required is a position vector and the tangent vectors and there is this parameter t_k which needs to be supplied.

For instance, the value of t_k can be chosen either considering a uniform parameterization. That is when each segment basically has got the parameterization between 0 and 1 and that is what we also call as the normalized splines. But we can also consider this parameterization being computed using chord length. What do we mean by chord length? It is basically the distance between the two data points which we are interpolating. It is a Euclidean distance between the two successive or the pair of points which we want to interpolate.

(Refer Slide Time: 06:30)



So we just take this distance measure and define that as the parameter t_k . If you look at from the point of view of parameterization this is a more natural way of defining a parameterization because you are considering the data spread over the space. Parameterization in turn is capturing how you are going along the curve. So the chord length parameterization is in some sense making this measure of the distance which you are going along the curve. This is of more relevance when we talk about certain aspects of animation.

The input of position vectors and tangent vectors does influence the kind of spline you are going to construct. For instance the magnitude of tangent vector can also influence the shape of the curve. Here is an example. If I have the point P_1 here and a point P_2 here just consider a spline between two points. I specify the tangent vectors as P_1' P_2' .

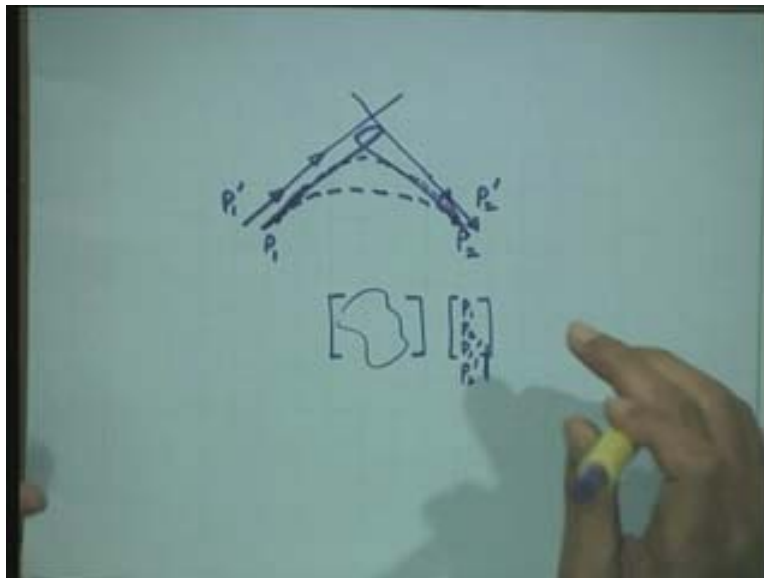
So, considering that they are in some sense the representative of the magnitude of these tangent vectors which I have drawn here you will get a curve of the kind like this. Now let us say I increase the magnitude of this tangent vectors so it goes there, it goes there and so on. What will happen is this curve would actually **lack** so you will have a curve like this. Remember what we have spline as basically a consequence of blending of the geometric information coming from the position vector and the tangent vectors so the magnitude is going to influence.

If you go back there is a matrix here and there is this P_1 P_2 P_1' P_2' these are not just directions but there is a magnitude involved in it and that is also going to scale the curve in some sense. You are going to blend this with this so this does affect. The circle is not the same way of parameterizing it. The circle has a parametric representation in terms of the angle you give. If I want to construct a circle using cubic spline I just cannot do it this way. This will not satisfy all the conditions of a circle and it may look

like a closed curve. This may not have all the conditions that a circle needs to satisfy. It will still satisfy the direction part of this tangent at the two points. But as I change the magnitude the curve seems to lack. That means it becomes sort of a loose curve and it is not a tight curve. And in fact it can go to the extent if I increase like this then this will actually go like this (Refer Slide Time: 13:26).

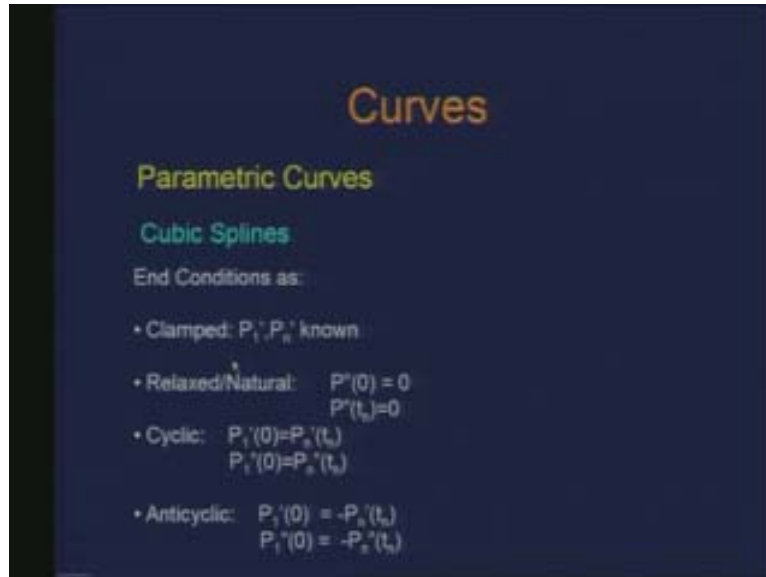
There is a loop, this is however not affecting its continuity the curve primes continuity is not getting affected. In fact just specifying the tangent vector is not sufficient to have the same curve you have to fix the magnitude in this representation of cubic splines. So basically what I am trying to demonstrate is that there are effects of the input you give in terms of this tangent vector.

(Refer Slide Time: 14:06)



We also observe that the end conditions which are specified also affect the shape of the curve. It is also a control in terms of the kind of spline you would like to obtain. For instance, we observed that these are the four commonly used end conditions. So, if the two end tangent vectors are specified we say that these are clamped end conditions and that is what we have basically studied so far.

(Refer Slide Time: 15:59)

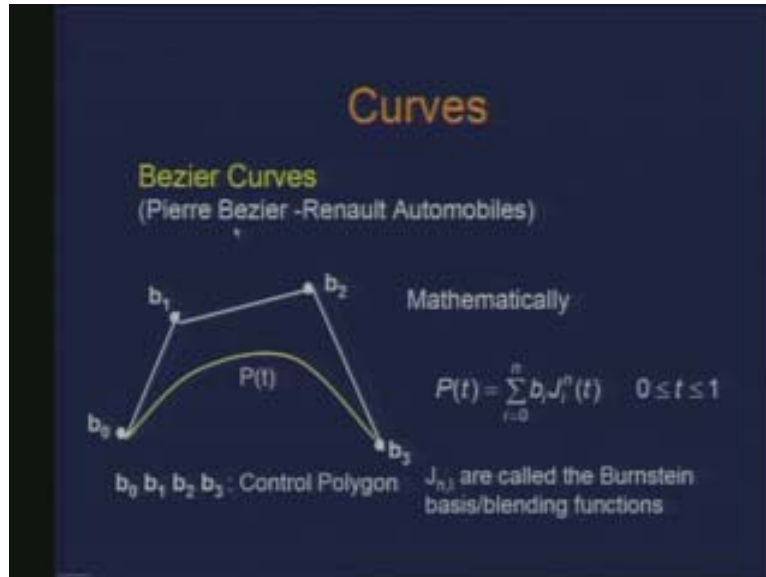


If we have the second derivative term at the two ends to be 0 then we have what we call as relaxed or natural end conditions. So remember that this is going to change the first and the last row of the matrix from where you would actually obtain the respective derivatives. And we also had seen the cyclic end conditions where we would like to have the two end tangent vectors and the second derivatives at the end to match. So this is particularly of relevance when we want to construct closed curves and there is also possibility that you have this condition as anticyclic and we saw the example of something like a racquet shape. Hence, these are kind of control handles by which you can change or control a shape of the spline you are constructing. They are always interpolating splines, they are interpolating the data points.

Now let us go to a different type of parametric curve. The different aspect which you want to see is where the user or the designer specifies a general shape through certain input for what he or she wants to obtain as the shape of the curve. So the user may be interested in specifying certain points. He may not be interested that the curve needs to pass through three these points whereas the curve needs to have certain control from this input or the shape has to be governed by that input.

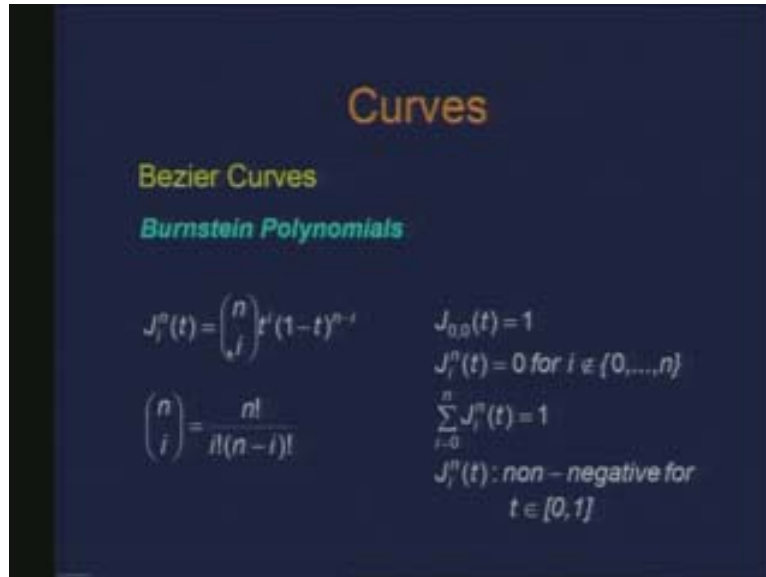
So, one such kind of curves is the Bezier curves. And in fact it was due to a person called Pierre Bezier in France who is basically working in Renault automobiles and his interest was for CAD CAM design to be able to have these free form curves which are easy to specify as far as the input is concerned and that is why the name Bezier is due to him. So what do we have here? We have the input given as points b_0 b_1 b_2 b_3 and this is the input given.

(Refer Slide Time: 18:42)



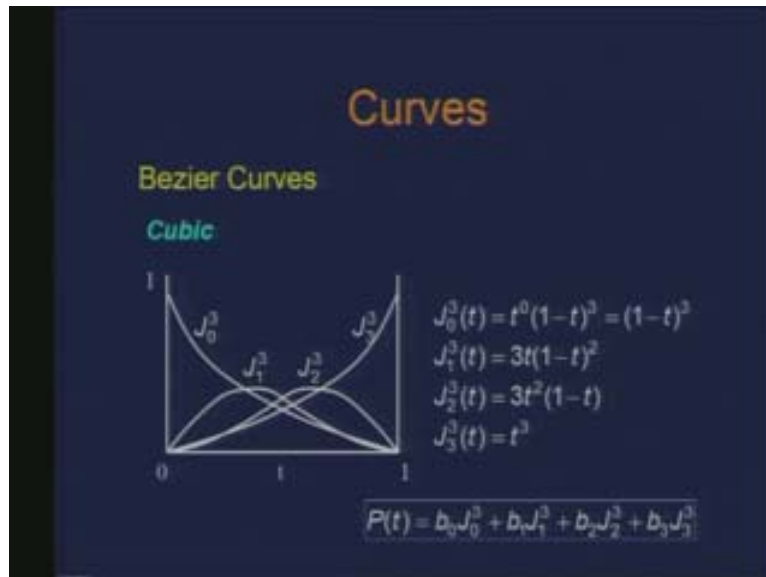
And the intention is that we would like to obtain a curve which is in some sense capturing the shapes specified by these points. There is this curve which is obtained because of the influence of these points. So again mathematically when you want to see it, those are the input points which are specified and some blending functions which are used for taking the combination of these points. These polynomials or the blending functions in this case are the Bernstein basis of blending functions and the points which are specified here are called as the control points and if I try to build sort of a polygon through the sequence of points I get a Bezier polygon or a control polygon. So the name control is given because I am controlling in some sense the shape of the curve which is going to be generated. Here again you have this parametric definition coming through a parameter defined between a range 0 and 1. The Bernstein polynomials are the blending functions in this case and this is how the Bernstein polynomials look like.

(Refer Slide Time: 21:07)



So this is ncr which is this n factorial divided by i factorial n minus i factorial. Now if we look at these polynomial if I want to evaluate at 0 and 0 so n equal to 0 i equal to 0 I have this equal to 1. And also for all i other than between 0 0 and n I have this equals to 0. So this gives me the definition span of the polynomial. And I also observe that it sums to 1, summation jn_i(t) for i equal to 0 to n sums to 1. So this is also a property which is called as partition of unity or unity partition. And I also observe that each of these jn_i(t) is non negative for the values for t between 0 and 1. Some of these are of relevance. Now if you take the example of cubic where the degree of Bernstein polynomial which I am considering is 3 then this is how the curve looks like. This is j₃₀, j₃₁, j₃₂, j₃₃ these are just the plots of these.

(Refer Slide Time: 23:25)



And when I define my parametric curve which is the Bezier curve in this case would look like this. Basically combinations of these polynomials with the Bezier control points look like this (Refer Slide Time: 24:15). So again I can write it in a form which I am familiar with which we did for the case in cubic spline where I take all these terms separate from the control points or the geometric information which is in this case control points then I can see a matrix which I can separate it again in the monomial form.

(Refer Slide Time: 24:19)

Curves

Bezier Curves

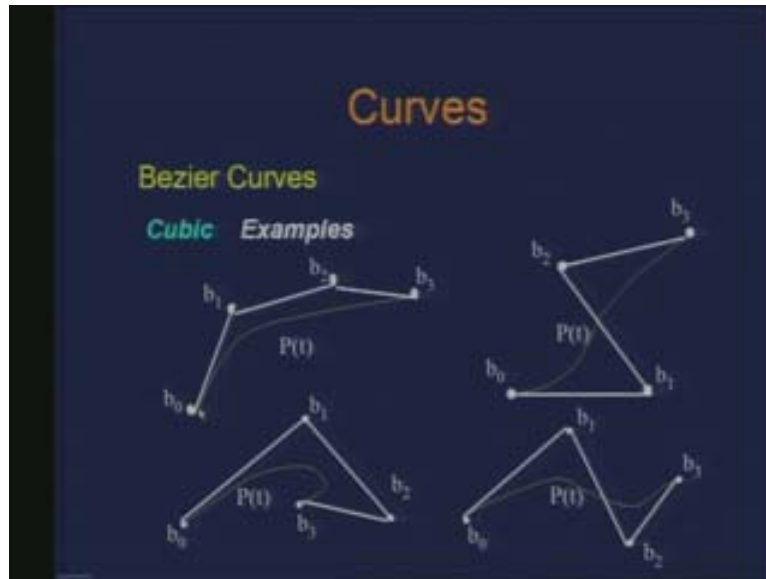
Cubic

$$P(t) = \begin{bmatrix} (1-t)^3 & 3t(1-t)^2 & 3t^2(1-t) & t^3 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

$$= \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

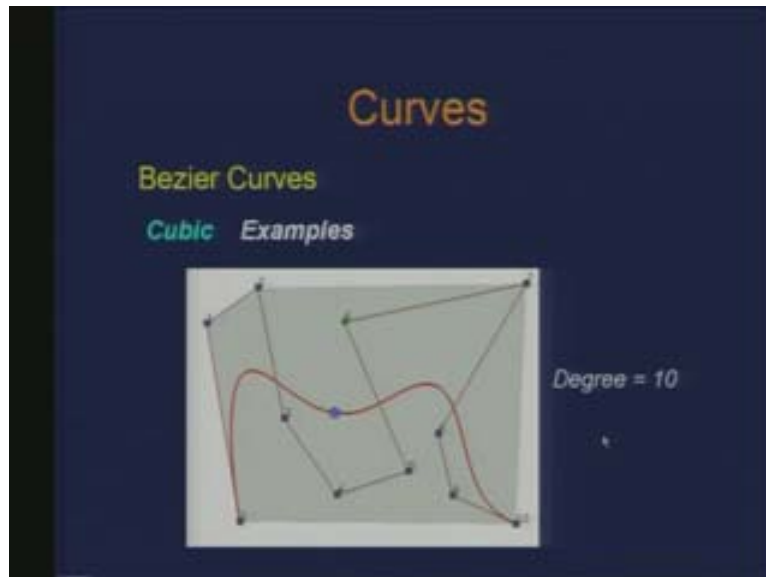
This is the monomial form of the coefficients $1t^2$ and t^3 so I get a matrix representation for cubic Bezier curve. Now these are the examples. So you have b_0 b_1 b_2 b_3 and this is the corresponding Bezier curve.

(Refer Slide Time: 25:13)



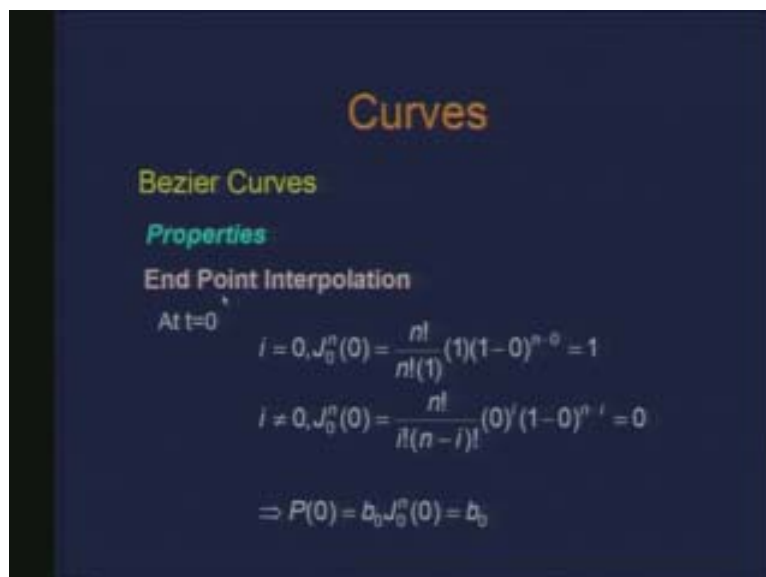
In this case this is b_0 b_1 b_2 b_3 and the curve is like this. By the way these are not exactly computed Bezier curves. **First these control polygons, so these are** definitely not the exact curves there is a possibility of error so these are not the computed curves. All you observe here is that these control points are in some sense giving the idea of the shape of the curve and that is what we were trying to achieve. And from the users prime perspective it becomes easier just to specify the location of these points and obtain the result in curve. So, again this is a little bigger curve and the other thing which we note is that there is a direct correlation between the number of control points of the Bezier polygon to the degree of the Bezier curve which I am designing. Here this was a cubic curve. (Refer Slide Time: 26:45) The number of control points I had was 4 and the degree of the curve is 3. Similarly, here I have eleven control points going from 0 and 10 so the degree of the curve is 10.

(Refer Slide Time: 27:13)



Therefore, degree and the number of control points are directly related. Now let us try to see certain properties of these Bezier curves. The first thing we observe just by the plots of various curves we have seen is that the Bezier curve interpolates to the end points. The first control point and the last control point it passes through.

(Refer Slide Time: 28:33)



That can easily be seen here. If I just substitute for t equal to 0 what happens to the j and i ? Here I see that it becomes 1 and the rest becomes 0. So it immediately gives us P at 0 as this and all the other terms are cancelled and I get this as b_0 . It is just a simple substitution.

Similarly at t equal to 1 if I substitute for the evaluation of j and n I observe that for i equal to n this turns out to be 1 and for the rest turns out to be 0. So again at p equal to 1 all I get is this which is b_n the last Bezier point.

(Refer Slide Time: 28:49)

Curves

Bezier Curves

Properties

End Point Interpolation

At $t=1$

$$i = n, J_n^n(1) = \frac{n!}{n!(1)} (1)^n (0)^{n-n} = 1$$

$$i \neq n, J_i^n(1) = \frac{n!}{i!(n-i)!} (1)^i (1-1)^{n-i} = 0$$

$$\Rightarrow P(1) = b_n J_{n,n}(1) = b_n$$

The next property which we are going to look at is the affine invariance.

(Refer Slide Time: 29:25)

Curves

Bezier Curves

Properties

Affine Invariance

Applying an affine transformation to the curve is equivalent to applying the transformation to the control points.

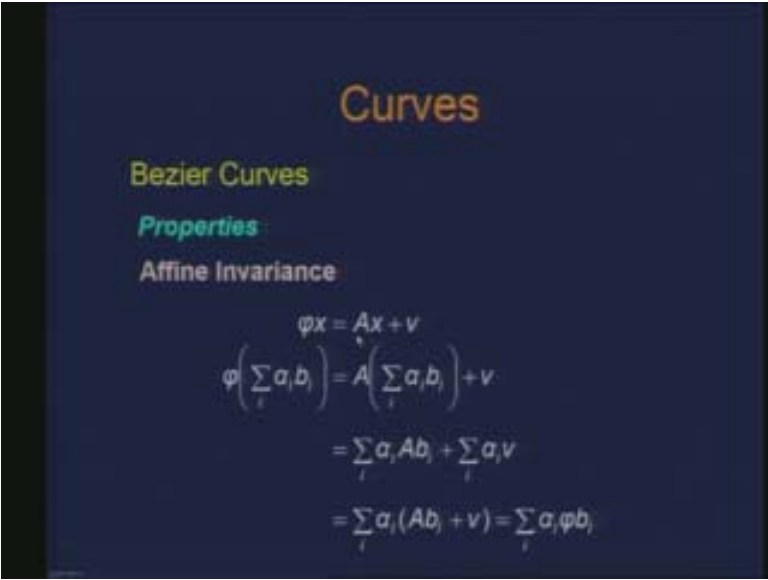
The diagram illustrates the affine invariance property. On the left, a Bezier curve $P(t)$ is shown with control points b_0, b_1, b_2, b_3 . An arrow points to the right, where the transformed curve is shown with transformed control points b'_0, b'_1, b'_2, b'_3 .

Affine invariance basically says that if I apply affine transformation to the curve it is actually equivalent to applying the transformation to the control points.

So basically what I mean here is that if this is the Bezier curve obtained by these control points b_0, b_1, b_2, b_3 and I apply some affine transformation to this so the curve would have gone there. Now instead if I had applied the affine transformation only to the control points which would have given me the control points as b_0, b_1, b_2, b_3 here then the curve computed from them would have been the same. Therefore that is a useful property because if you want to apply affine transformation to the curve you rather apply it to the control points.

How can we see it more mathematically? If I consider an affine map which I define as ϕ to some x , it basically means that I have some matrix A which is a 3 into 3 matrix in 3D space or 2 into 2 matrix in 2D and there is this translation term here which is some V . So this x in our case which is the curve $P(t)$ is something which looks like this. There are these Bezier control points and there are some coefficients coming from the Bernstein polynomials. Now when I apply this to this it basically means this on the side and all I need to do is just substitute for x this and I just rewrite this where I take this A inside just as the linear operator and knowing that in our case the summation to the coefficients is equal to 1.

(Refer Slide Time: 30:55)



Curves

Bezier Curves

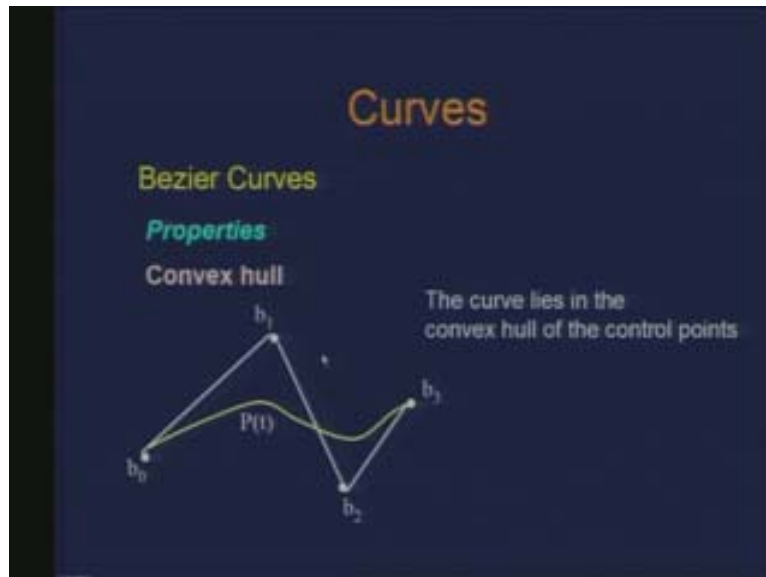
Properties:

Affine Invariance

$$\begin{aligned}\phi x &= Ax + v \\ \phi \left(\sum_i \alpha_i b_i \right) &= A \left(\sum_i \alpha_i b_i \right) + v \\ &= \sum_i \alpha_i A b_i + \sum_i \alpha_i v \\ &= \sum_i \alpha_i (A b_i + v) = \sum_i \alpha_i \phi b_i\end{aligned}$$

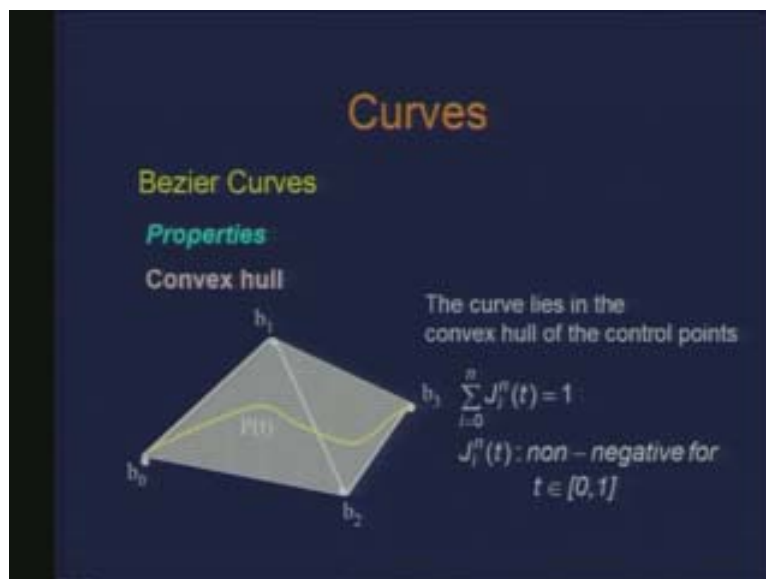
We saw that the summation of Bernstein polynomials is actually 1 so I just use that multiply this by unity which means summation of α is 1 and just rearrange the terms in this way which gives me this which is to say that this is in affine map of b_i primes or an affine transformation of b_i . So I can also see the same thing mathematically. The other property which is also of a great relevance is that the curve lies in the convex hull of the control points.

(Refer Slide Time: 33:31)



If I have this basic curve here defined as $P(t)$ coming from the control points b_0, b_1, b_2, b_3 then this curve is going to reside inside the convex hull made by these points b_0, b_1, b_2, b_3 . This means first of all let us try to construct the convex hull of the control points which looks like this, this is the convex hull, all it says that the curve is going to lie within this.

(Refer Slide Time: 33:43)

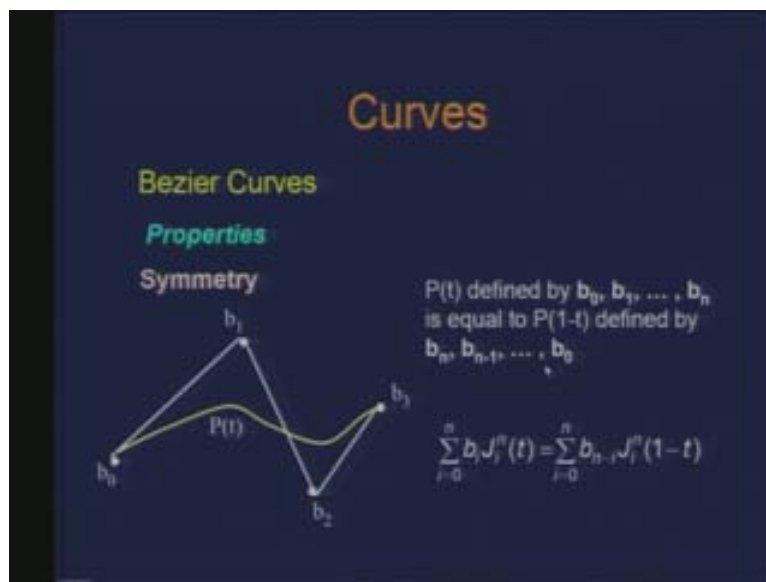


Now we will use the properties of our Bernstein polynomials. If I have to have a point or a curve within the convex hull of these control points would actually mean that the point which is computed using the convex combination of those set of points or the control points. And how do I get these convex combinations? It is by these two properties.

Remember that each of them individually is non negative and then they sum to 1. This gives me the convex hull property. Where else can you use this property?

Here is an example, you have this curve or anything designed using these curves. So, if you want to determine an intersection or you are trying to determine whether the object which is constructed using this curve collides with any other object what can you do there is you can basically perform the intersection with respect to the convex hull first and only when the convex hull intersects you need to find the intersection with the curve. Therefore it is called pruning. You can prune out intersection by performing the intersection with convex hull and that is the advantage you get. There is another property which is kind of obvious. it is symmetric with respect to t and 1 minus t meaning that if I have $P(t)$ defined by the Bezier points b_0, b_1, b_n equal to the curve $P(1-t)$ defined by b_n, b_{n-1}, \dots, b_0 which is actually from here.

(Refer Slide Time: 37:19)



Hence this is saying that this curve is the same as this curve and that is easy to establish. So far what we have done is we have basically looked at the curves which are defined through the parameter t in an interval 0 and 1 . But there could be instances of situations where this parameter definition is not between 0 and 1 so the question is does it matter? All we are saying is that in that case I need a transformation. So this t was basically defined between 0 and 1 , there is another parameter u which is defined between a and b . Thus, I just need to map this using some affine transformation, using just the ratios and substitute for this. So practically it does not really matter even if I have the parameter defined in a different span than 0 and 1 . I can always map that to the parameter between 0 and 1 .

(Refer Slide Time: 37:50)

Curves

Bezier Curves

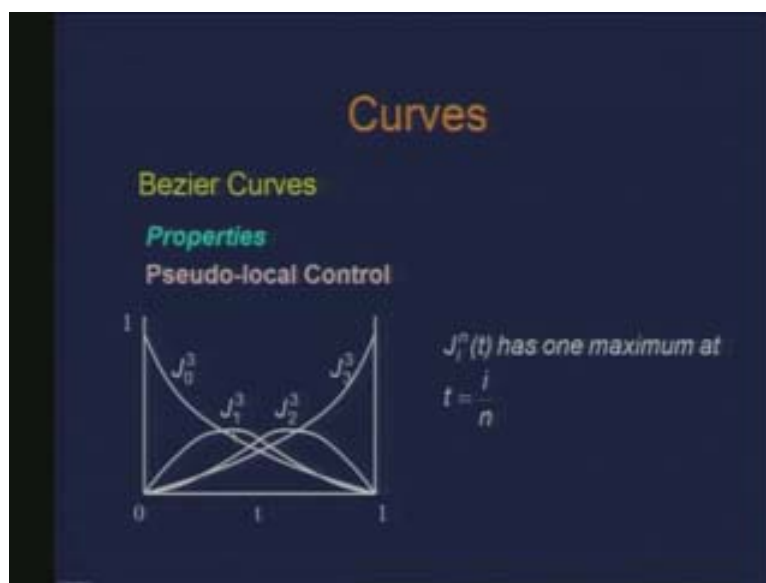
Properties

Domain Parameter Transformation

$$\begin{aligned} t &\in [0, 1] \\ u &\in [a, b] \\ t &= \frac{u - a}{b - a} \end{aligned} \quad \cdot \quad \sum_{i=0}^n b_i J_i^n(t) = \sum_{i=0}^n b_i J_i^n\left(\frac{u - a}{b - a}\right)$$

There is another property something like a pseudo local control. When you look at these Bernstein polynomials here they are for a cubic case, what we observe is that this $J_i^n(t)$ each one of them has a maximum given at t equal to i by n . For instance, for this you have the maximum here, for this you have a maximum somewhere there, for this you have a maximum somewhere there so it is $2i + 1$ by n and it is somewhere here.

(Refer Slide Time: 39:38)

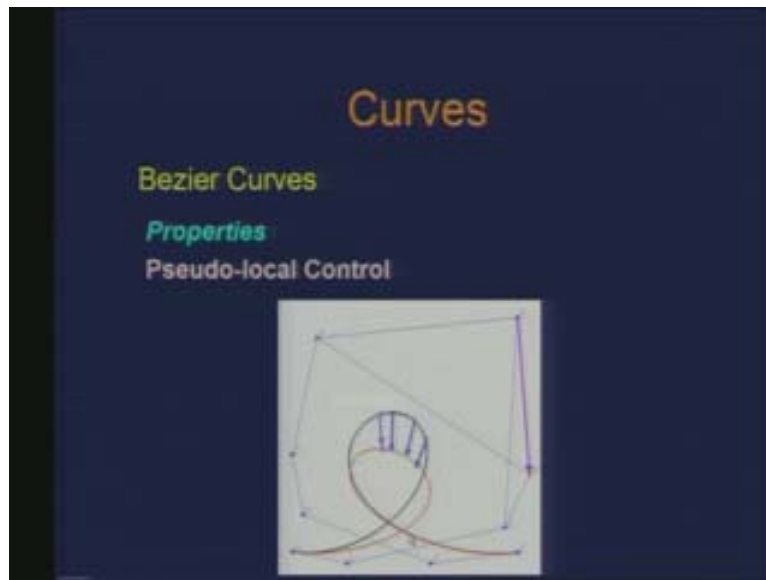


Basically it means that if I move a particular control point of the Bezier polygon then what is the kind of influence I have to the curve?

In some sense it is telling you that if I move the Bezier point b_2 the maximum influence is going to be there around this parameter t equal to $2/3$. In some sense it is telling you the kind of a local control with respect to each of the control points I have. Just by the definition of the Bezier curve we observe that moving any point of the Bezier polygon is going to change the entire curve. But there is some sort of a local influence of a particular control point which in some sense tells you that there is a local control although there is a change of the entire curve.

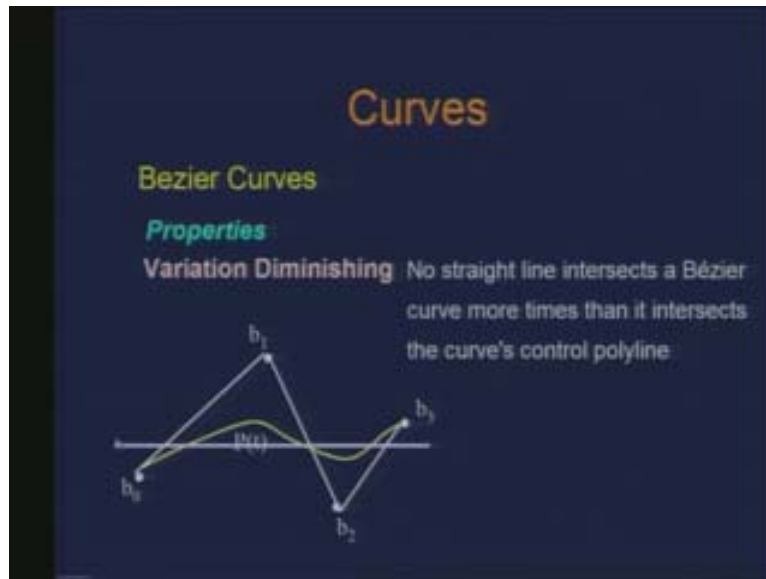
In fact this aspect of global control or non local control of Bezier curve is sometimes considered as a limitation to the design of parametric curves because every time you change the curve the entire curves changes. Here is an example, if this point is moved to this point the curve which was initially this changes to this. And you see that the entire curve is changing but there is a larger change to the neighborhood of the parameter influenced by this point which is $1/n$.

(Refer Slide Time: 42:38)



So this is important to know from the point of view of designing curve. So as a user you should know the range of influence to the curve when I am going to change a particular point, **this is useful from the interface.** Another property is variation diminishing. This is for the case of planar curves. When I talk about space curve they become plain. So here it says, for planar curve no straight line intersects a Bezier curve more number of times than it intersects the curves control polyline.

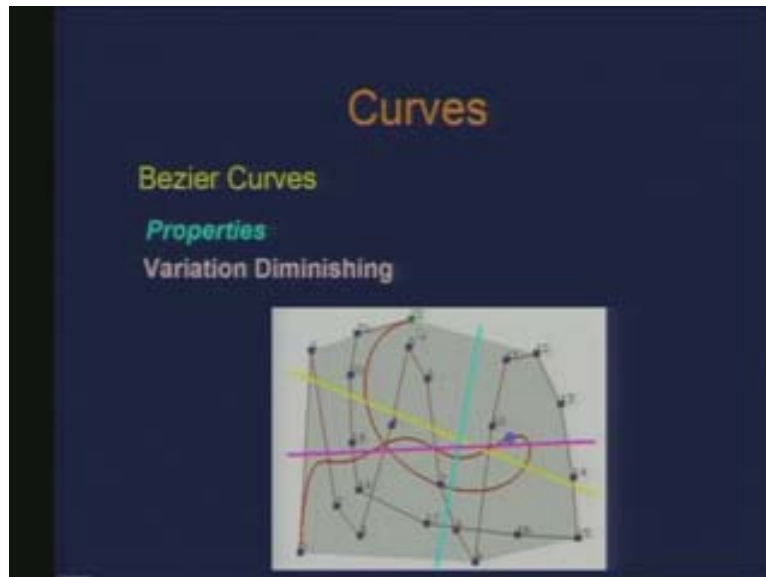
(Refer Slide Time: 43:50)



In this case these are the legs of the polylines on the Bezier control point or control polygon and this is the curve. So when I look at the intersection with respect to this line I observe that it intersects at three points and so does the curve. But this is the maximum number of times a straight line can intersect with the curve which in turn says that the kind of twist and turns offered by the various segments of the control polygon is so sort of a maximum. The curve may follow it but it will be less than that. So the variations are diminishing.

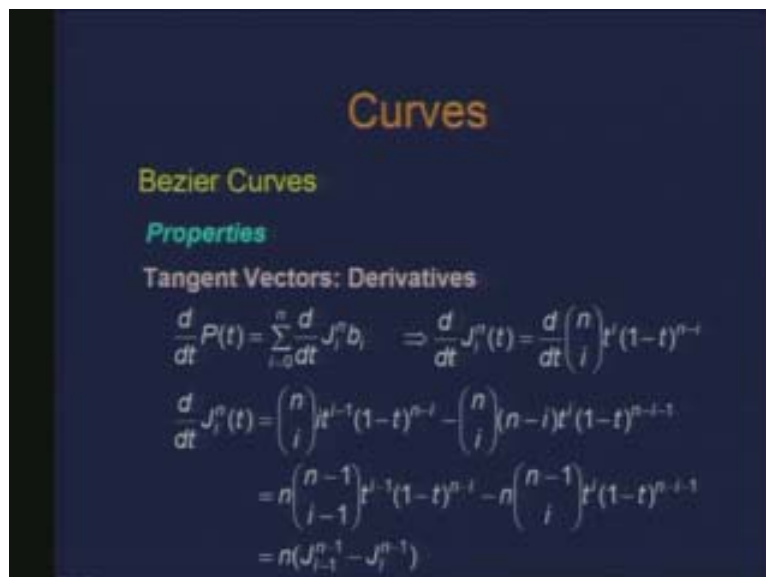
And again if you go back and look at the convex hull property, so if I have the Bezier control polygon convex maximum it is going to intersect is two times, a line can intersect two times and that is what I will have with the curve as maximum. So it sort of asserts the convexity. Here we have a curve with twenty three control points so it is of degree 22 so at any point for instance if I consider the yellow line it intersects one two three four five six seven times whereas to the curve it intersects one two and three times.

(Refer Slide Time: 45:39)



Now we come to the tangent vectors and the derivatives which actually give us the tangent vectors for the curve. So if I am interested in finding out the derivative of the curve which is just the meaning that I differentiate with respect to the parameter t in turn means that I need to do a differentiation of these jni primes. So now let us just examine the differentiation of these jni primes.

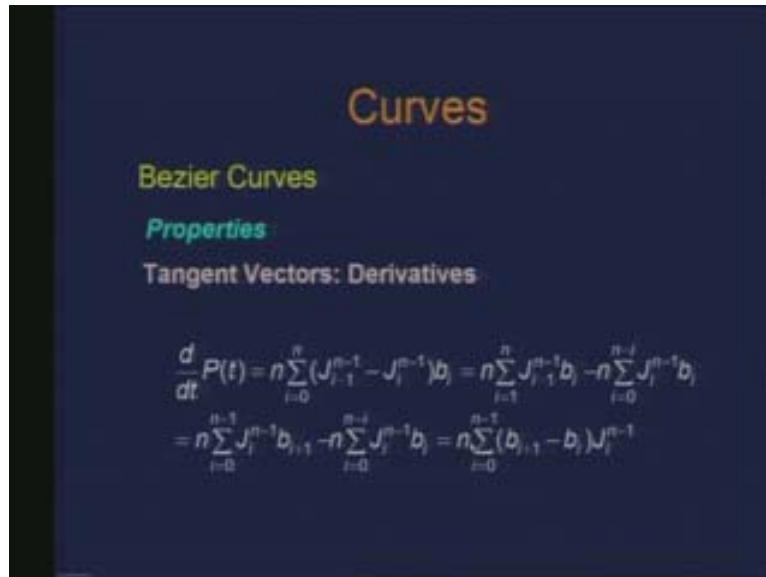
(Refer Slide Time: 46:50)



So if I differentiate this which is basically this I get something like this; it to the power i minus 1 (i minus t) to the power n minus i and the second term is this which I can again rearrange and write in this form and eventually I get something like n times j minus 1 i

minus 1 minus jn minus 1 i. So now when I see the derivative of the curve itself I use that which is this multiplied to the Bezier points bi primes.

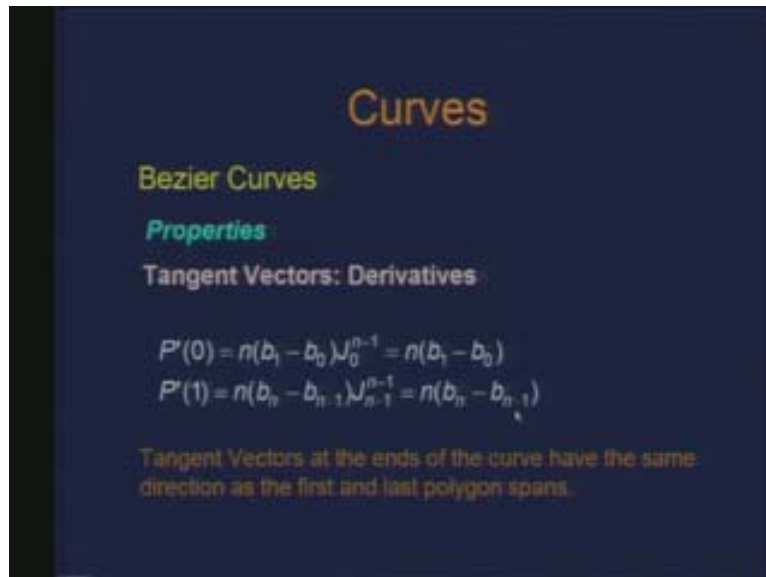
(Refer Slide Time: 48:04)



And again I do some rearrangement of the terms and eventually I get this which is n times summation i 0 to n minus 1 bi plus 1 minus bi jn minus 1i. So all I am doing is some kind of a first difference of bi primes. Now if I try to evaluate these at two end points I observe this which is n times b1 minus b0 and at the other end it is n times bn minus 1 bn minus 1 which is saying that the tangent vectors at the ends of the curve has the same direction as the first and last polygon spans.

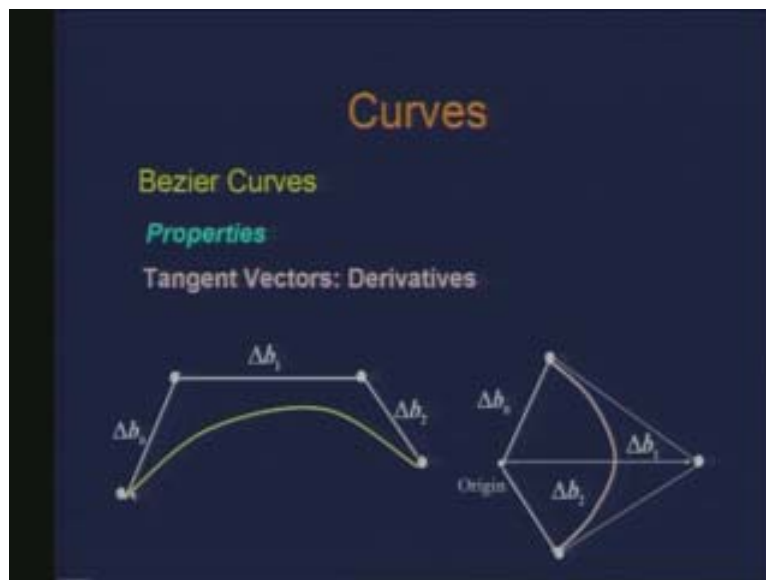
Now you can do a correlation of the cubic splines as well. I can map these two if I want to. There I was specifying the two end tangent vectors and the points so here if I had given the four control points I am basically doing a similar thing, it is equivalence in some sense. In this case I am basically computing through this n. So if I have a cubic so this is three times the first leg and this is again three times the last leg.

(Refer Slide Time: 50:12)



Now again let us say I take the case of a cubic Bezier curve where this is my Bezier polygon and I just basically write the first differences as these legs of the polygon. This is delta b₀, this is delta b₁ and this is delta b₂.

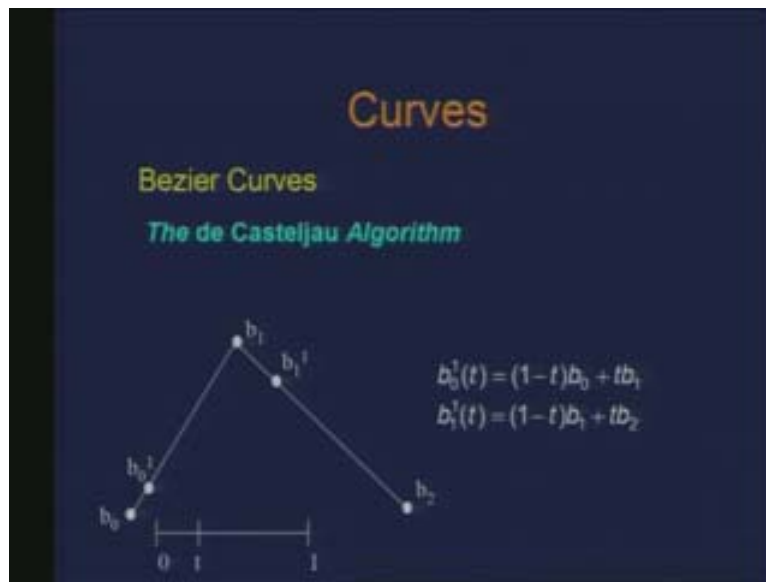
(Refer Slide Time: 50:27)



Now remember that if I go back here this is my derivative for the curve which is again taking the first difference and having this Bernstein polynomial of degree n minus 1. Now if you look at this, what do you say about the result? What is Bezier curve? This is nothing but another Bezier curve obtained by this difference. So this is directly what I have tried to demonstrate here. The only thing is since they are the first differences of the

points they are not defined in the Euclidean space at these vectors so I need to define some more region somewhere fix it somewhere here and just plot the first differences here as vectors. So this gives me a point here, this gives me another point here and this gives me another point here and these are the three Bezier points. And if I plot now the Bezier curve obtained by these three points that is what my tangent vectors are. So the first derivative of the Bezier curve is actually a Bezier curve. Now we move on to another type of construction of Bezier curves. So far what we have seen is basically a basis of blending functions used as the Bernstein polynomials for obtaining the Bezier curves.

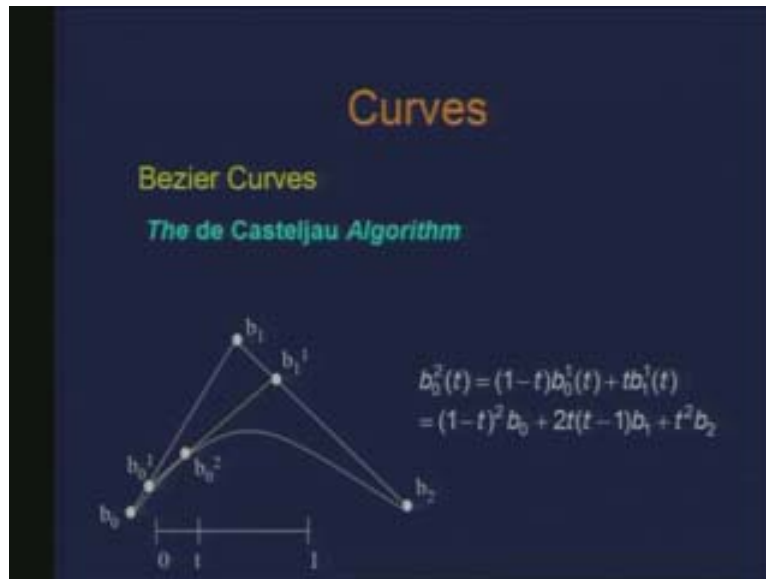
(Refer Slide Time: 53:09)



Now I have the case of a quadratic Bezier curve which needs for me to define three Bezier points b_0 , b_1 and b_2 and I have this parametric domain defined to t between 0 and 1. Now what do I do for a given t where I want to evaluate or construct a point on the Bezier curve? I do a linear interpolation for the legs of the polygon. So I locate this point b_0^1 as a linear interpolation of b_0 and b_1 through the parameter t which locates the point here. Similarly, for the other span I do a linear interpolation between b_1 and b_2 which locates the point b_1^1 . And then I do the same linear interpolation through the parameter t for the leg b_0^1 and b_1^1 .

Basically I do a linear interpolation between these two points and obtain the point b_2^0 . So b_2^0 is a linear interpolation of this. If you recall, this was a linear interpolation of this leg and this was the linear interpolation along this leg. And if I just open these up this is what I get, and what are these? These are the Bernstein polynomials of degree 2. So what do I get is a Bezier curve. So this b_2^0 point which I obtain after the second interpolation is actually a point on the curve.

(Refer Slide Time: 54:07)



So this construction is due to the person called de Casteljau who gave this algorithm of constructing parametric curves which were later on called as Bezier curves because Bezier came only in 1970s whereas de Cateljau came in early 1900s. So he had already done this. So the same construction using Bernstein polynomial actually can be obtained by a repetitive linear interpolation.