Curves

Eri Prasetyo

Curves before computers

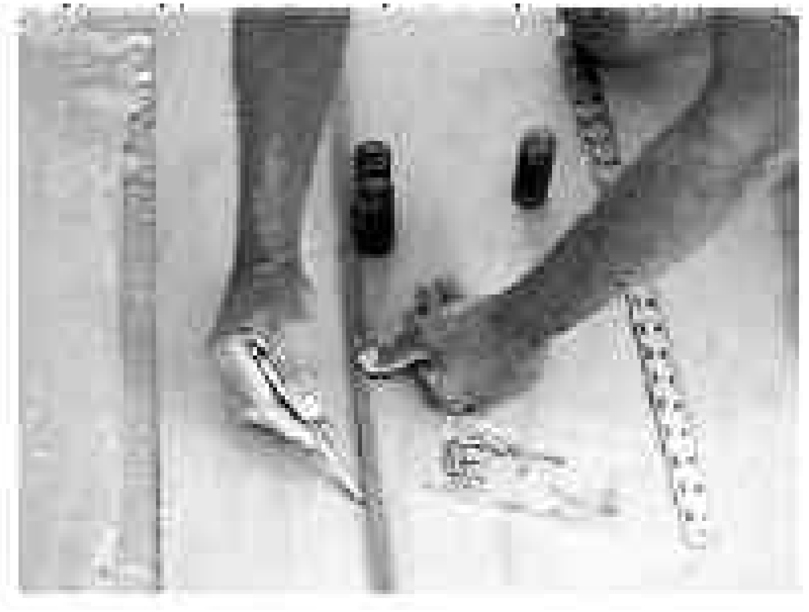
The loftsman’s spline :

• Long, narrow strip of wood or metal

• Shaped by lead weights called ‘ducks’

• Gives curves with second-order continuity, ussually

 Used for designed cars, ships, airplane, etc.



Motivation for Curves

What do we use curves for ?

- Building models

- Movement paths

- Animation

Mathematical Curve Representation

- Explicit = y = f(x)
- Implicit f(x,y,z) = 0

  hard to work with
- parametric (f(u)g(u))

Parametric Polynomial Curves

We'll use parametric curves where the function are all

Polynomials in the parameter.

$$x(u) = \sum_{k=0}^{n} a_k u^k$$

$$y(u) = \sum_{k=0}^{n} b_k u^k$$

Advantages :

- Easy (efficient) to compute
- Infinitely differentiable

Cubic Curves

Fix n = 3

For simplicity we define each cubic function within the range

$$0 \le t \le 1$$

$$\mathbf{Q}(t) = \begin{bmatrix} x(t) & y(t) & z(t) \end{bmatrix}$$

$$Q_x(t) = a_x t^3 + b_x t^2 + c_x t + d_x$$
$$Q_y(t) = a_y t^3 + b_y t^2 + c_y t + d_y$$
$$Q_z(t) = a_z t^3 + b_z t^2 + c_z t + d_z$$

Compact Representation

Place all cooficients into a matrix

$$\mathbf{C}=\begin{bmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \\ d_x & d_y & d_z \end{bmatrix} \quad \mathbf{T}=\begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix}$$

$$Q(t)=\begin{bmatrix} x(t) & y(t) & z(t) \end{bmatrix}=\begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix}\begin{bmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \\ d_x & d_y & d_z \end{bmatrix} = \mathbf{T}\cdot\mathbf{C}$$

$$\frac{d}{dt}Q(t)=Q'(t)=\frac{d}{dt}\mathbf{T}\cdot\mathbf{C}=\begin{bmatrix} 3t^2 & 2t & 1 & 0 \end{bmatrix}\cdot\mathbf{C}$$

Constraining the cubies

Redefine C as a product of the basis matrix M and the

4 element column vector of constraints or geometry vector G

C = M.G

$$\mathbf{Q}(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ m_{41} & m_{42} & m_{43} & m_{44} \end{bmatrix} \begin{bmatrix} G_{1x} & G_{1y} & G_{1z} \\ G_{2x} & G_{2y} & G_{2z} \\ G_{3x} & G_{3y} & G_{3z} \\ G_{4x} & G_{4y} & G_{4z} \end{bmatrix}$$

$$= \mathbf{T} \cdot \mathbf{M} \cdot \mathbf{G}$$

Hermite Curves

Determined by

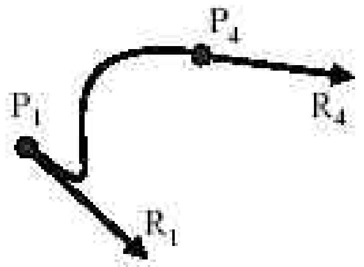
- endpoints P1 and P4
- Tangent vectors at the endpoints R1 and R4

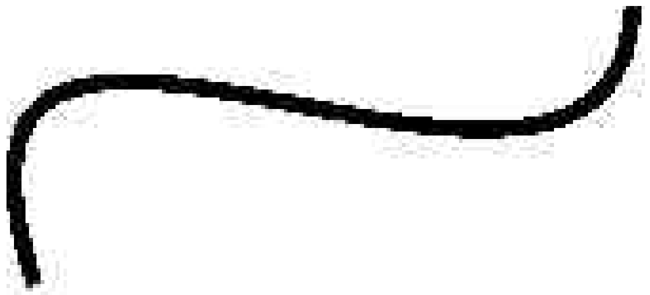
So

$$\mathbf{Q}(t) = \mathbf{T} \cdot \mathbf{M}_h \cdot \mathbf{G}_h$$

Where

$$\mathbf{G}_h = \begin{bmatrix} P_{1x} & P_{1y} & P_{1z} \\ P_{4x} & P_{4y} & P_{4z} \\ R_{1x} & R_{1y} & R_{1z} \\ R_{4x} & R_{4y} & R_{4z} \end{bmatrix}$$





Computing Hermite basis matrix

The constraints on Q(0) and Q(1) are found by direct

Subtitution :

$$Q(0) = \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} \cdot \mathbf{M}_h \cdot \mathbf{G}_h$$

$$Q(1) = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix} \cdot \mathbf{M}_h \cdot \mathbf{G}_h$$

Tangents are defined by

$$Q'(t) = \begin{bmatrix} 3t^2 & 2t & 1 & 0 \end{bmatrix} \cdot \mathbf{M}_h \cdot \mathbf{G}_h$$

so constraints on tangents are:

$$Q'(0) = \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix} \cdot \mathbf{M}_h \cdot \mathbf{G}_h$$

$$Q'(1) = \begin{bmatrix} 3 & 2 & 1 & 0 \end{bmatrix} \cdot \mathbf{M}_h \cdot \mathbf{G}_h$$

Computing Hermite basis matrix

Collecting all constraints we get

$$\begin{bmatrix} P_{1x} & P_{1y} & P_{1z} \\ P_{4x} & P_{4y} & P_{4z} \\ R_{1x} & R_{1y} & R_{1z} \\ R_{4x} & R_{4y} & R_{4z} \end{bmatrix} = \mathbf{G}_h = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} \mathbf{M}_h \cdot \mathbf{G}_h$$

So

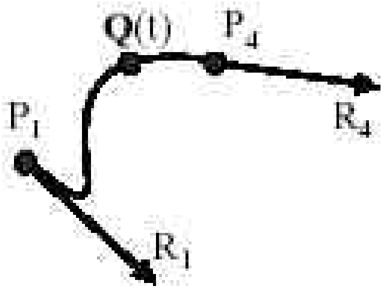
$$\mathbf{M}_h = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix}^{-1} = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Computing a Points

Given two endpoints (P1,P4) and two endpoints tangents vectors

(R1,R4) :

So



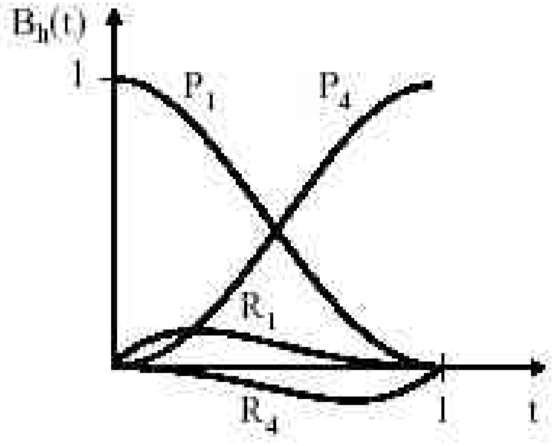
$$\mathbf{Q}(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{P}_1 \\ \mathbf{P}_4 \\ \mathbf{R}_1 \\ \mathbf{R}_4 \end{bmatrix}$$

Blending Function

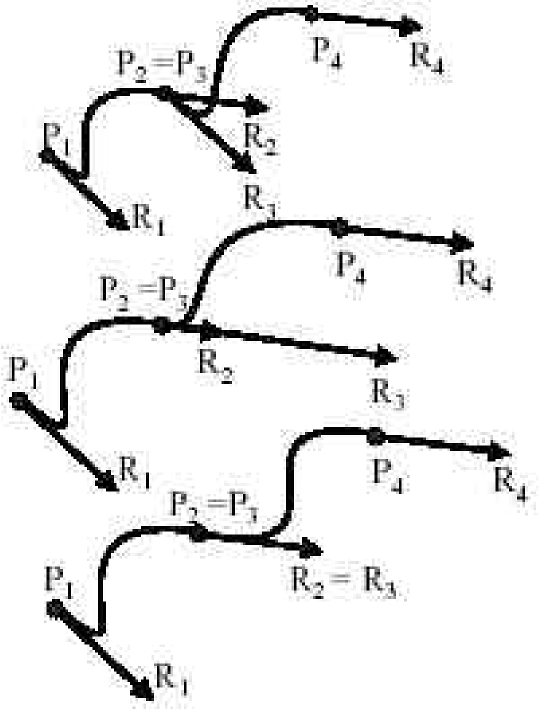
Polynomials weighting each element of geometry vector

$$\mathbf{Q}(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{P}_1 \\ \mathbf{P}_4 \\ \mathbf{R}_1 \\ \mathbf{R}_4 \end{bmatrix}$$

$$= \mathbf{B}_h(t) \begin{bmatrix} \mathbf{P}_1 \\ \mathbf{P}_4 \\ \mathbf{R}_1 \\ \mathbf{R}_4 \end{bmatrix}$$



Continuity of Splines



C^0: points coincide,
velocities don't

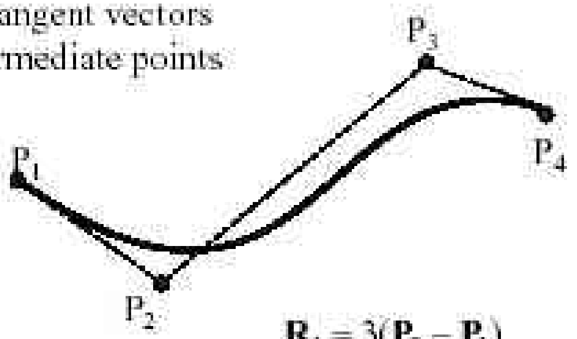
G^1: points coincide,
velocities have same direction

C^1: points and velocities
coincide

Q: What's C^2?

Bezier Curves

Indirectly specify the tangent vectors by specifying two intermediate points



$$\mathbf{R}_1 = 3(\mathbf{P}_2 - \mathbf{P}_1)$$

$$\mathbf{R}_1 = 3(\mathbf{P}_4 - \mathbf{P}_3)$$

$$\mathbf{G}_b = \begin{bmatrix} P_{1x} & P_{1y} & P_{1z} \\ P_{2x} & P_{2y} & P_{2z} \\ P_{3x} & P_{3y} & P_{3z} \\ P_{4x} & P_{4y} & P_{4z} \end{bmatrix} = \begin{bmatrix} \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \\ \mathbf{P}_4 \end{bmatrix}$$

Bezier basis matrix

Establish the relation between the Hermite and Besier geometry vectors:

$$\mathbf{R}_1 = 3(\mathbf{P}_2 - \mathbf{P}_1)$$
$$\mathbf{R}_1 = 3(\mathbf{P}_4 - \mathbf{P}_3)$$

$$\mathbf{G}_h = \begin{bmatrix} \mathbf{P}_1 \\ \mathbf{P}_4 \\ \mathbf{R}_1 \\ \mathbf{R}_4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -3 & 3 & 0 & 0 \\ 0 & 0 & -3 & 3 \end{bmatrix} \begin{bmatrix} \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \\ \mathbf{P}_4 \end{bmatrix} = \mathbf{M}_{bh} \mathbf{G}_b$$

Bezier basis matrix

$$\begin{aligned}\mathbf{Q}(t) &= \mathbf{T}\cdot\mathbf{M}_h\cdot\mathbf{G}_h = \mathbf{T}\cdot\mathbf{M}_h\cdot(\mathbf{M}_{hb}\cdot\mathbf{G}_b) \\ &= \mathbf{T}\cdot(\mathbf{M}_h\cdot\mathbf{M}_{hb})\cdot\mathbf{G}_b = \mathbf{T}\cdot\mathbf{M}_b\cdot\mathbf{G}_b\end{aligned}$$

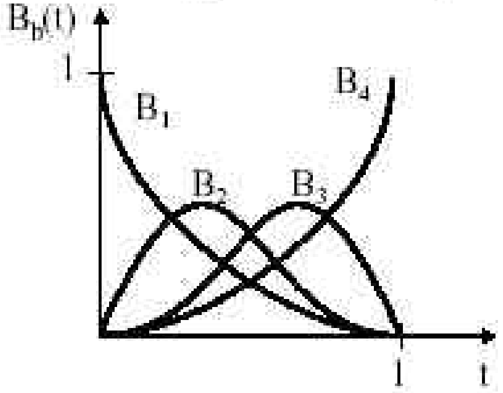
$$\mathbf{M}_b = \mathbf{M}_h\mathbf{M}_{hb} = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{Q}(t) = \mathbf{T}\cdot\mathbf{M}_b\cdot\mathbf{G}_b$$

Bazier Blending function

a.k.a. Bernstein polynomials

$$\mathbf{Q}(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \\ \mathbf{P}_4 \end{bmatrix} = \mathbf{B}_b(t) \begin{bmatrix} \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \\ \mathbf{P}_4 \end{bmatrix}$$



Alternative Bezier Formulation

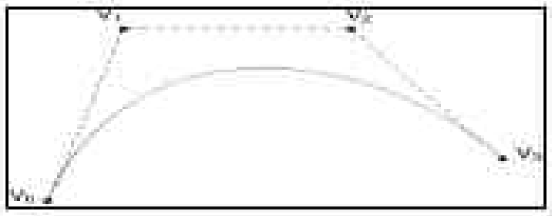
$$Q(t) = \sum_{i=0}^{3} P_i \binom{3}{i} t^i (1-t)^{3-i}$$

$$\mathbf{Q}(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \end{bmatrix}$$

Displaying Bezier Curves

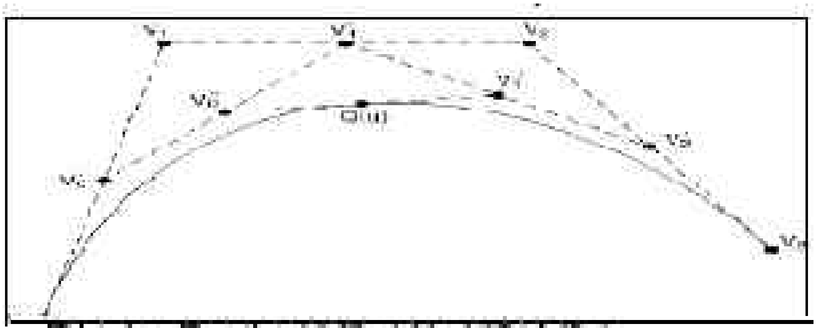
How could we draw one of these things?

```
DisplayBezier( V0, V1, V2, V3 )
begin
    if ( FlatEnough( V0, V1, V2, V3 ) )
        Line( V0, V3 );
    else
        do something smart;
end;
```



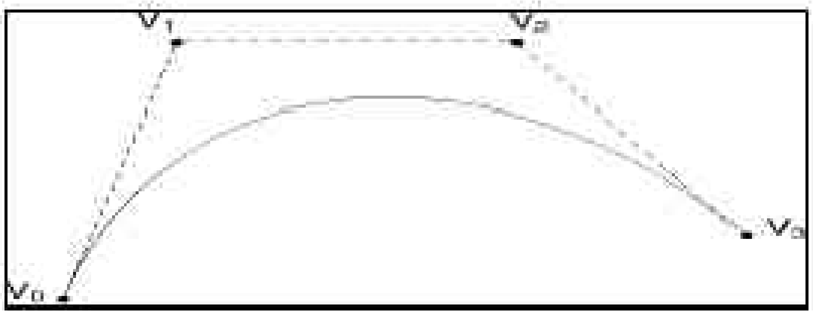
It would be nice if we had an *adaptive* algorithm, that would take into account flatness.

Subdivide and conquer



```
DisplayBezier( V0, V1, V2, V3 )
begin
    if ( FlatEnough( V0, V1, V2, V3 ) )
        Line( V0, V3 );
    else
        Subdivide(V) ⇒ L, R
        DisplayBezier( L0, L1, L2, L3 );
        DisplayBezier( R0, R1, R2, R3 );
end;
```

Testing for Flatness



Compare total length of control polygon to length of line connecting endpoints:

$$\frac{|V_0 - V_1| + |V_1 - V_2| + |V_2 - V_3|}{|V_0 - V_3|} < 1 + \varepsilon$$

More Complex Curves

Suppose we wants to draw a more complex curve

Why not use a high-order bezier ?

Instead , we'll splice together a curve from individual segments that are cubic Bezier ?

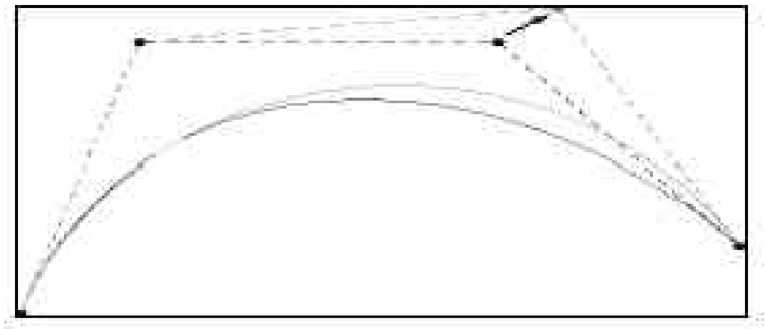
Why Cubic ?

There are three properties we'd like to have in our newly constructed splines …………

Local Control

One problem with Bezier is that every control point affects every point on the curve (except the endpoints)

Moving a single control points affects the whole curve

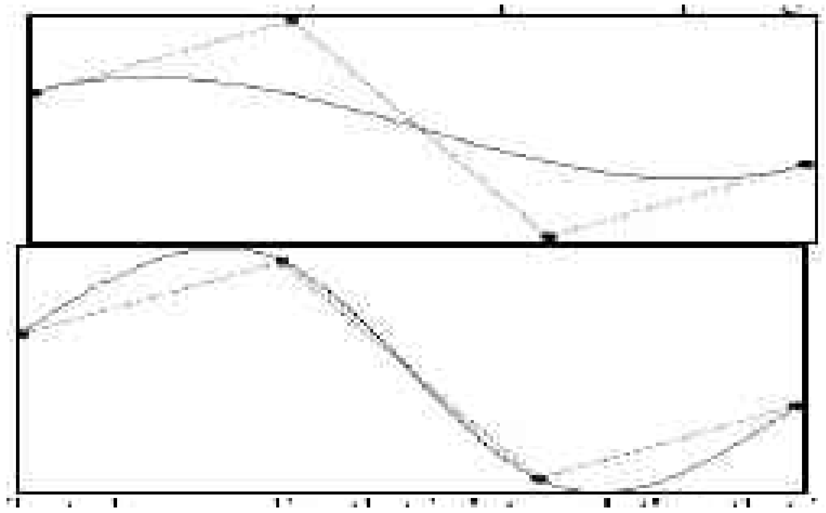


We'd like our spline to have local control, that is, have each control point

Affect some well-defined neighborhood around that point.

Interpolation

Bezier curves are approximating. The curve does not pass throught all the control points. Each point pulls the curve toward it, but other points are pulling as well.

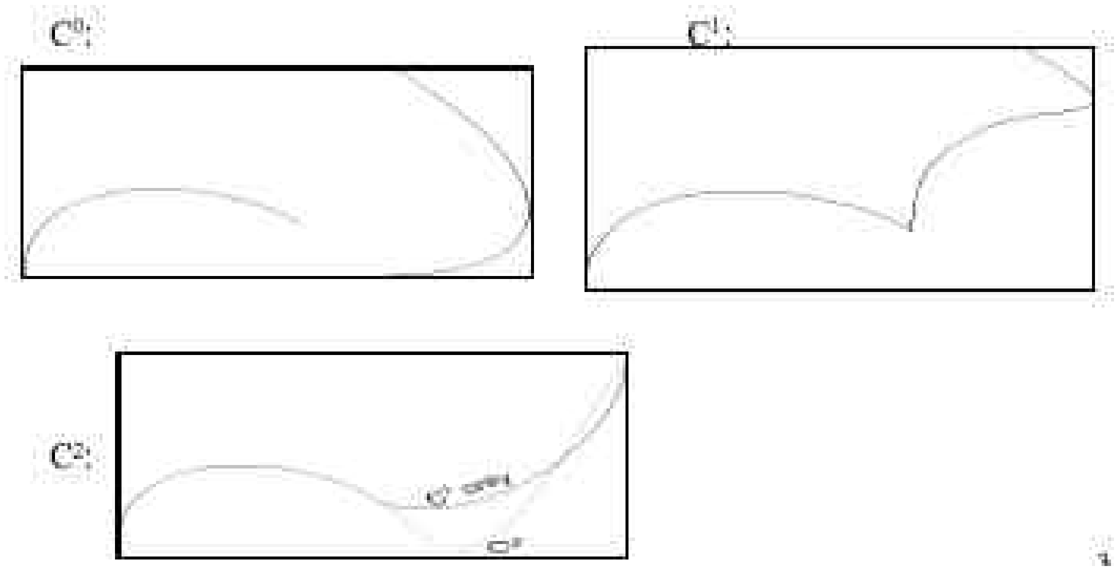


We’d like to have a spline that is interpolating, that is that always passes through every control point.

Continuity

We want our curve to have continuity. There shouldn't be an abrupt change when we move from one segment to the next.

There are nested degrees of continuity



Ensuring continuity

Let’s look continuity first

Since the functions defining a bezier curve are polynomial,

All their derivatives exist and are continous.

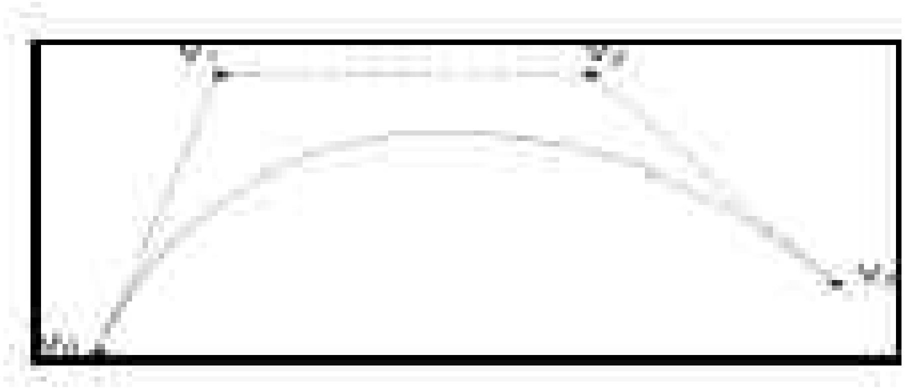
Therefore, we only need to worry about the derivatives at the endpoints of the curve.

First, we’ll rewrite our equation for Q(t) in matrix form :

$$Q(u) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & \\ -3 & 3 & & \\ 1 & & & \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix}$$

Derivatives at the endpoints

$$Q'(0) = 3(V_1 - V_0)$$
$$Q'(1) = 3(V_3 - V_2)$$
$$Q''(0) = 6(V_0 - 2V_1 + V_2)$$
$$Q''(1) = 6(V_1 - 2V_2 + V_3)$$

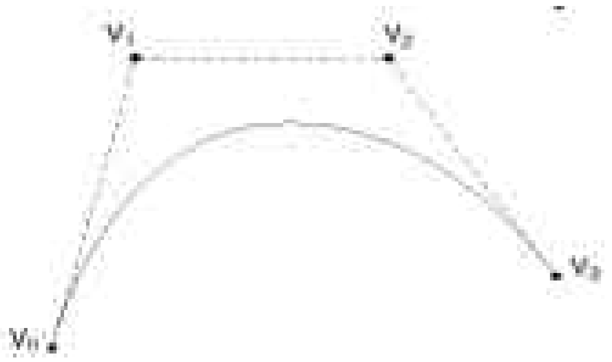


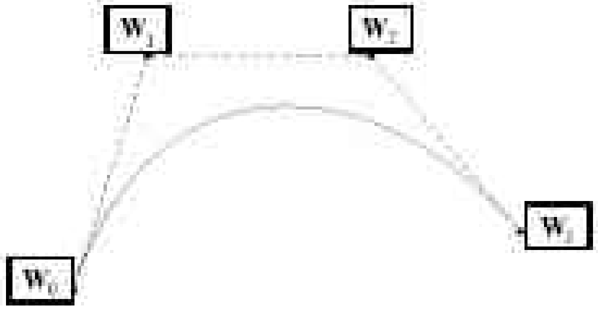
In general, the n derivatives at an endpoint depends only on the n+1 points nearest that endpoint.

Ensuring C^2 continuity

Suppose we have a cubic bezier defined by (V0,V1,V2,V3), and we want to attach another curve (W0,W1,W2,W3) to it.

So that there is C^2 continuity at the joint.



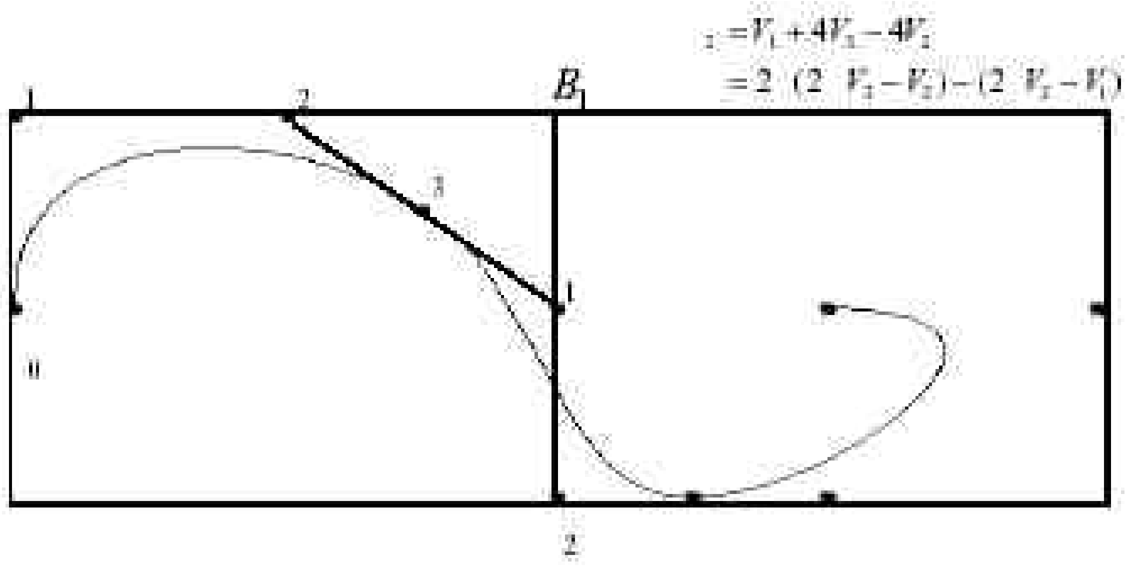


$$V_3 = W_0$$
$$V_3 - V_2 = W_1 - W_0$$
$${}_1 - 2V_2 + V_3 = W_0 - 2W_1 + W_2$$
$$W_2 = V_1 + 4V_3 - 4V_2$$

A-Frames and continuity

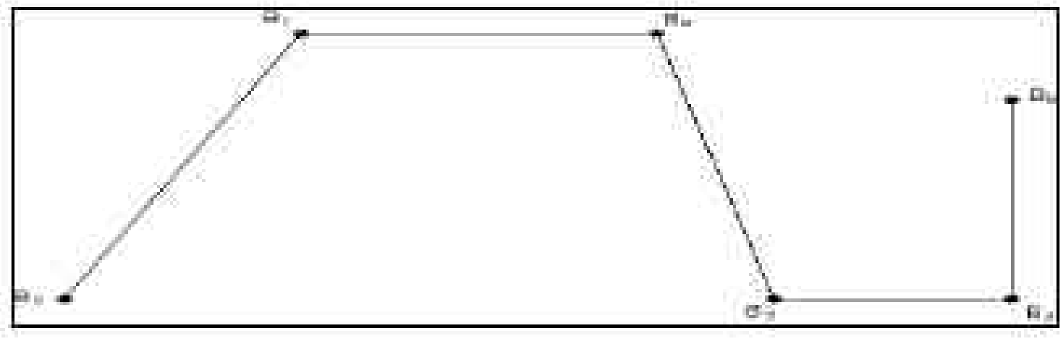
Let’s try to get some geometrical intuition about what this last continuity equation means.

If a and b are points, what is (2a – b) ?



Building a complex spline

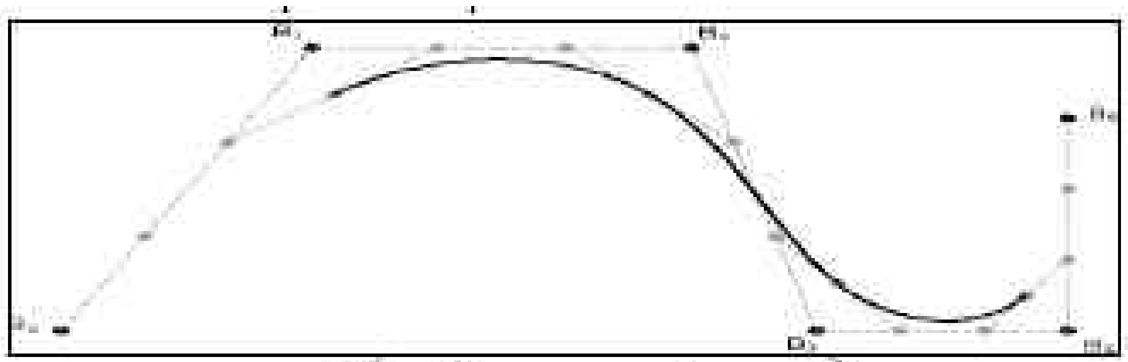
Instead of specifying the bezier control points themselves, let's specify the corners of the A-frames in order to build a C^2 continous spline.



These are called B-splines. The strating set of points are called de Boor points.

B-Splines

Here is the completed B-spline



$${}_0 = \frac{1}{2}\left(B_0 + \frac{2}{3}(B_1 - B_0) + B_1 + \frac{1}{3}(B_2 - B_1) \right)$$

$$V_1 = B_1 + \frac{1}{3}(B_2 - B_1)$$

$${}_2 = B_0 + \frac{2}{3}(B_1 - B_0)$$

$$V_3 = \ldots$$

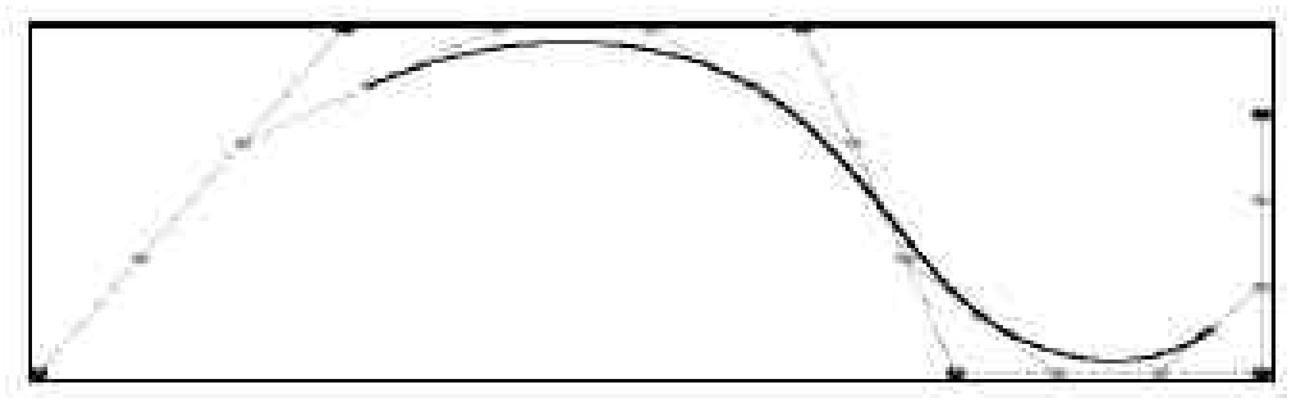
What are the bezier control points , in terms of the de Boor points ?

Endpoints of B-spline

We can see that B-splines don’t interpolate the de Boor points.

It would be nice if we could at least control the endpoints of the splines explicity.

There’s a hack to make the spline begin and at control points by repeating them.



B-Spline matrix

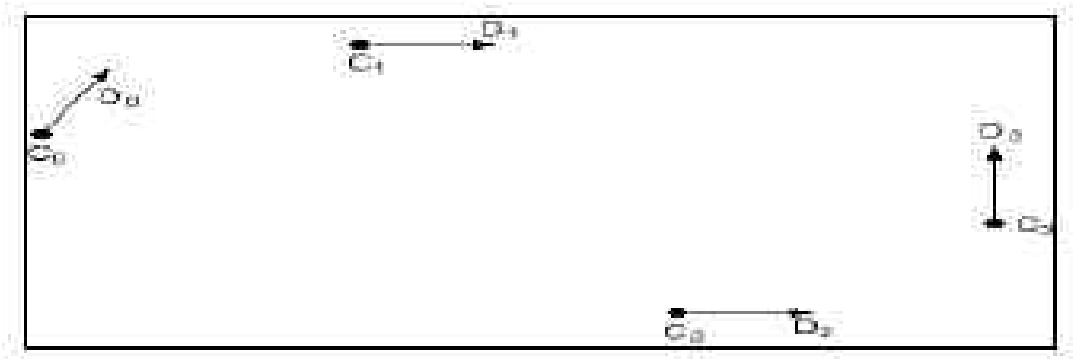
$$\mathbf{Q}(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \frac{1}{6} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \end{bmatrix}$$

C^2 interpolating splines

Interpolation is a really handy property to have

How can we keep the C^2 continuity we get with B-splines but get interpolation too ?

Here's the idea behind C^2 interpolation splines. Suppose we had cubic beziers connecting our control points C0,C1,C2,……….. And that we somehow knew the first derivative of the spline at each point



What are the V and W control points in term of Cs and Ds

Finding the derivatives

Now what we need to do is solve for the derivatives. To do this we'll use the C^2 continuity requirement.

$$
\begin{aligned}
{}_0 &= C_0 \\
V_1 &= C_0 + \tfrac{1}{3} D_0 \\
{}_2 &= C_1 - \tfrac{1}{3} D_1 \\
V_3 &= C_1
\end{aligned}
\qquad\qquad
\begin{aligned}
{}_0 &= C_1 \\
W_1 &= C_1 + \tfrac{1}{3} D_1 \\
{}_2 &= C_2 - \tfrac{1}{3} D_2 \\
W_3 &= C_2
\end{aligned}
$$

$$6\,(V_1 - 2V_2 + V_3) = 6\,(W_0 - 2W_1 + W_2)$$

Finding the derivatives,

Here's what we've got so far :

$$D_0 + 4D_1 + D_2 = 3(C_2 - C_0)$$
$$D_1 + 4D_2 + D_3 = 3(C_3 - C_1)$$
$$\mathrm{M}$$
$$D_{m-2} + 4D_{m-1} + D_m = 3(C_m - C_{m-2})$$

How many equation is this ?

How many unknowns are we solving for ?

Not quite done yet

We have two additional degrees of freedom, which we can nail down by imposing more conditions on the curve.

There are various ways to do this. We'll use the variant called **natural C^2 interpolating splines**, which requires the second derivative to be zero at the endpoints.

This condition gives us the two additional equations we need. At the C_0 endpoint, it is:

$$6(V_0 - 2V_1 + V_2) = 0$$

Solving for derivatives

Let’s collect our m+1 equations into a single linear system:

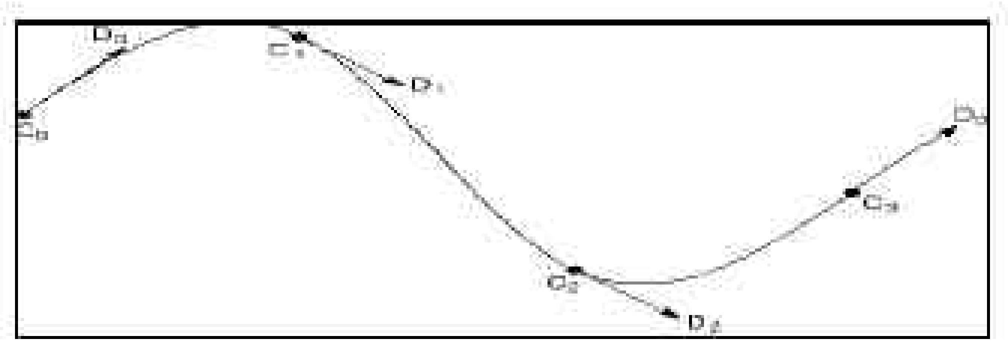
$$\begin{bmatrix} 2 & 1 & & & & \\ 1 & 4 & 1 & & & \\ & 1 & 4 & 1 & & \\ & & & \mathrm{O} & & \\ & & & 1 & 4 & 1 \\ & & & & 1 & 2 \end{bmatrix} \begin{bmatrix} D_0 \\ D_1 \\ D_2 \\ \mathrm{M} \\ D_{m-1} \\ D_m \end{bmatrix} = \begin{bmatrix} 3(C_1 - C_0) \\ 3(C_2 - C_0) \\ 3(C_3 - C_1) \\ \mathrm{M} \\ 3(C_m - C_{m-2}) \\ 3(C_m - C_{m-1}) \end{bmatrix}$$

It’s easy to solve than it looks

We can use forward elimination to zero out everything below the diagonal , then back substitution to compute each D value

C^2 interpolating spline

Once we've solved for the real Dis , we can plug them in to find our bezier control points and draw the final spline :



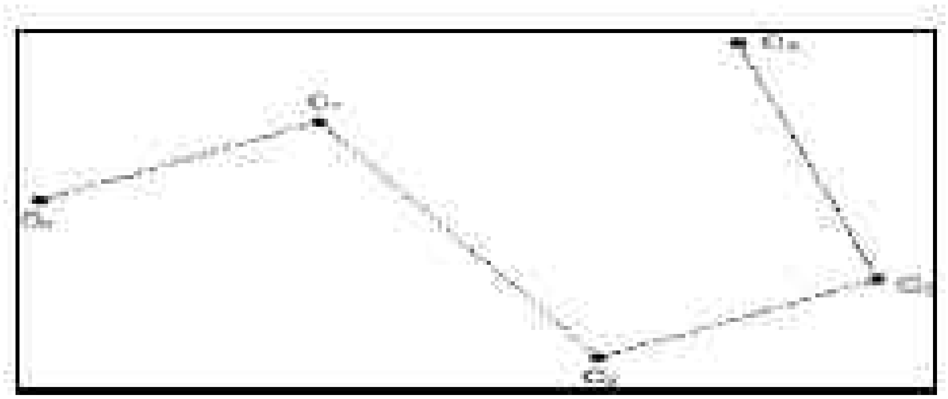
Have we lost anything ?

A third option

If we’re willing sacrifice C continuity , we can get interpolation and local control .

Instead of finding the derivatives by solving a system of continuity equations, we’ll just pick something arbitrary but local.

If we set each derivatives to be constant multiple of the vector between the provious and next controls, we get a catmull-rom spline.



Catmull-rom splines

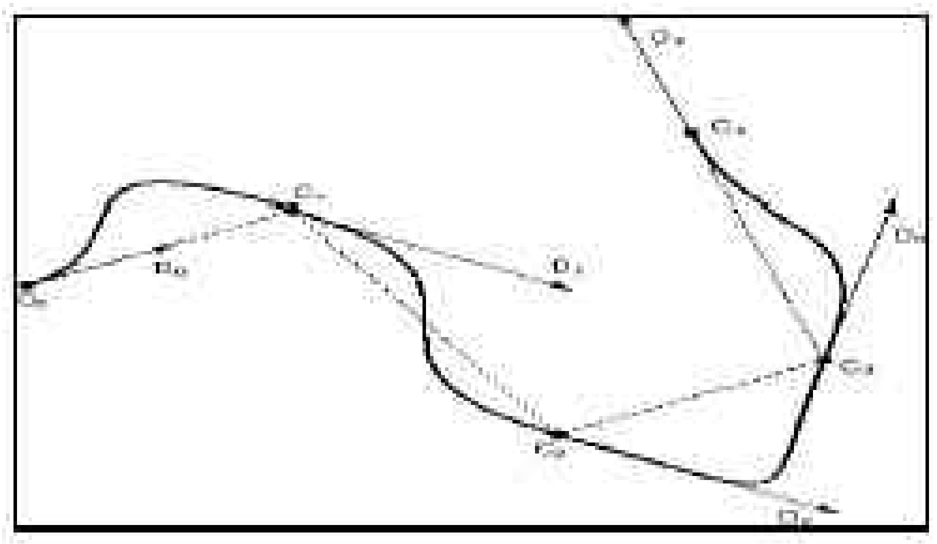
The math for catmull-rom splines is pretty simple :

$D0 = C1-C0$

$D1 = \frac{1}{2}(C2-C0)$

$D2 = \frac{1}{2}(C3-C1)$

$Dn = Cn-Cn-1$



Catmull-Rom basis matrix

$$\mathbf{Q}(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \frac{1}{2} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 2 & -5 & 4 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \end{bmatrix}$$