ARLAB

ME/CprE/ComS 557

Computer Graphics and Geometric Modeling

Modeling

December 3rd, 2015

Rafael Radkowski

IOWA STATE UNIVERSITY
OF SCIENCE AND TECHNOLOGY

COURSE ANNOUNCEMENT

ME 625
Surface Modeling

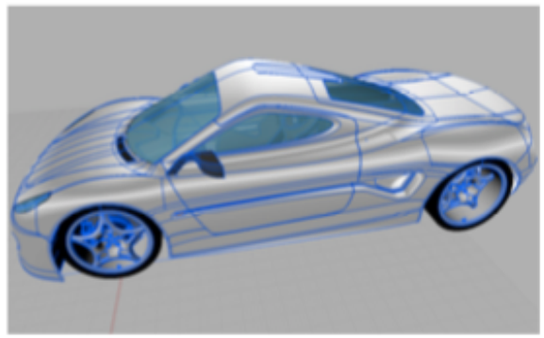
Spring 2016

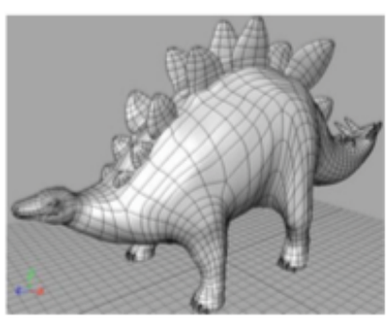
Instructor:
James Oliver

ARLAB

Surface modeling technologies are at the core of all contemporary computer shape modeling tools, spanning applications as diverse as mechanical, industrial, aerospace, naval, architectural, and even apparel design. This course explores the theory and practice of contemporary parametric sculptured surface design. Topics include:

- Curve and surface basics
- B-spline curves and surfaces
  - Basis functions
  - Derivatives
- Rational B-spline curves and surfaces
  - Conics
  - Re-parameterization
- Geometric tools
  - Knot insertion
  - Degree elevation
- Construction Techniques
- Trimmed surfaces
- Interpolation and Fitting
- Point Inversion and projection
- Shape modification
- Surface/surface intersection
- Applications – design, mesh generation, NC milling, integration with solid modeling systems, high-dimensional NURBS, others.





Text: *The NURBS Book*, by Les Piegl and Wayne Tiller, Springer-Verlag

Meeting Time: T, TH 11:00 – 12:15

Also available via Engineering/LAS Online Education

VRAC|HCI

IOWA STATE UNIVERSITY
OF SCIENCE AND TECHNOLOGY

Content

- Splines
- Terminology
- Cubic Spline
- Bezier Splines

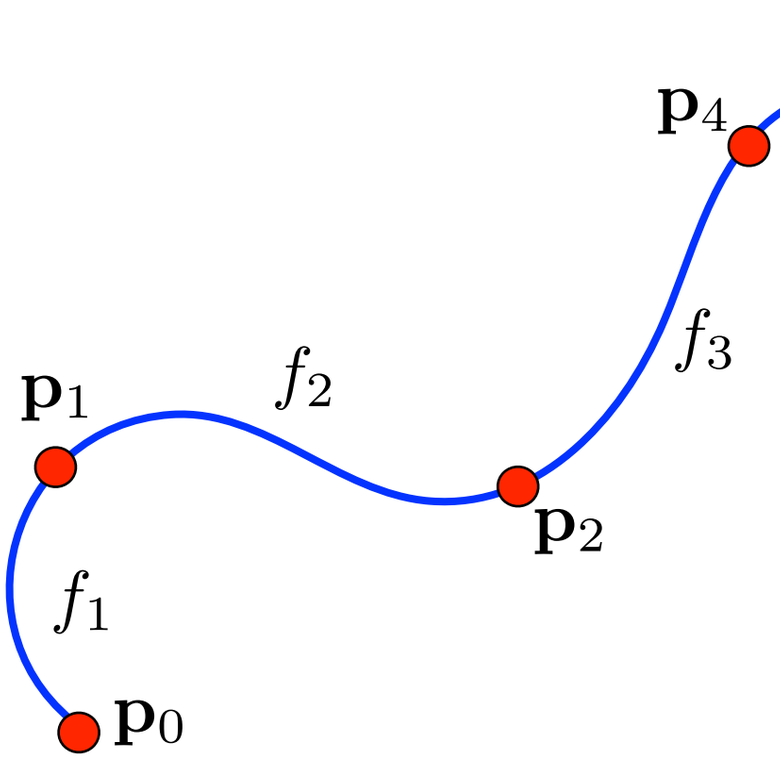
next time:

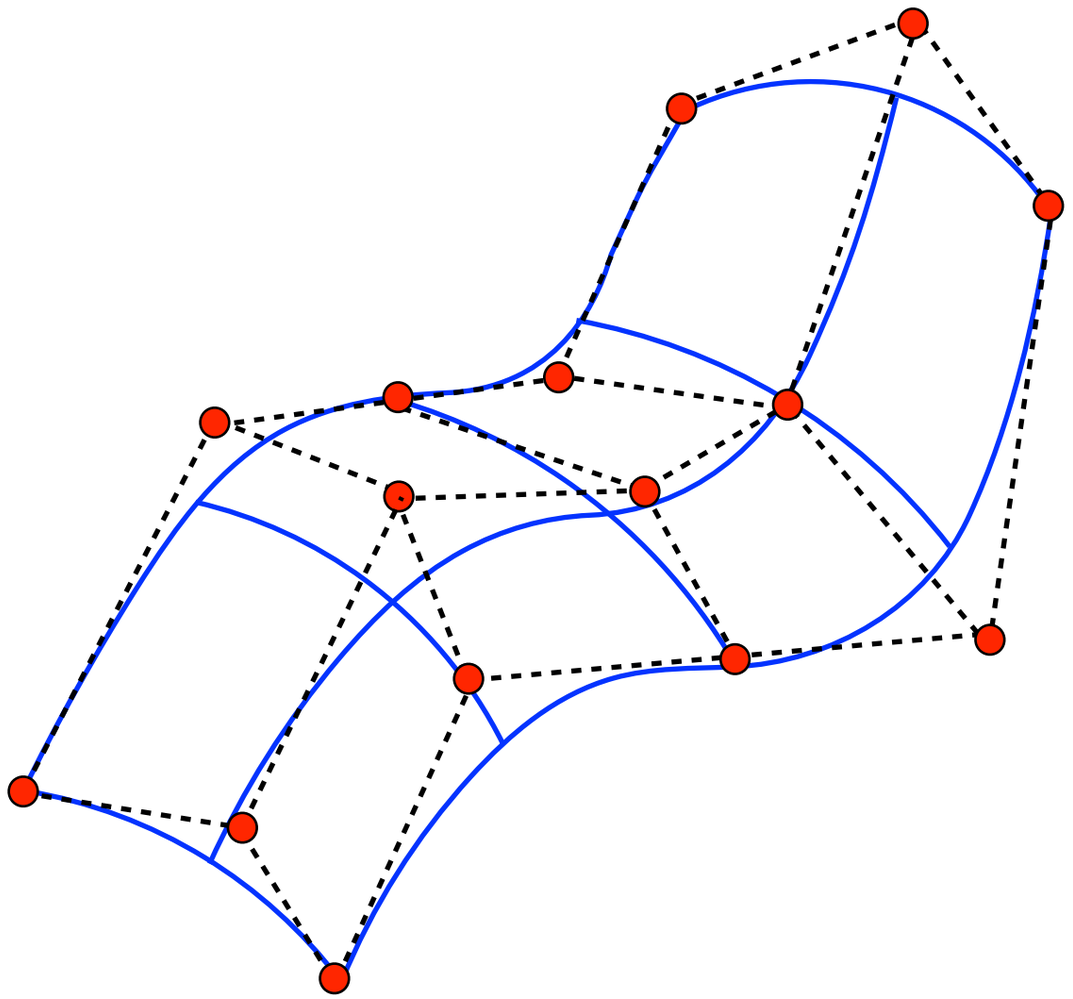
- surfaces
- drawing

Spline

In modeling terminology, a spline curve is a flexible strip used to produce a smooth curve through a designated set of points. Mathematically, we can describe this curve with piecewise cubic polynomial functions.

A spline surface can be described with two sets of orthogonal spline curves.



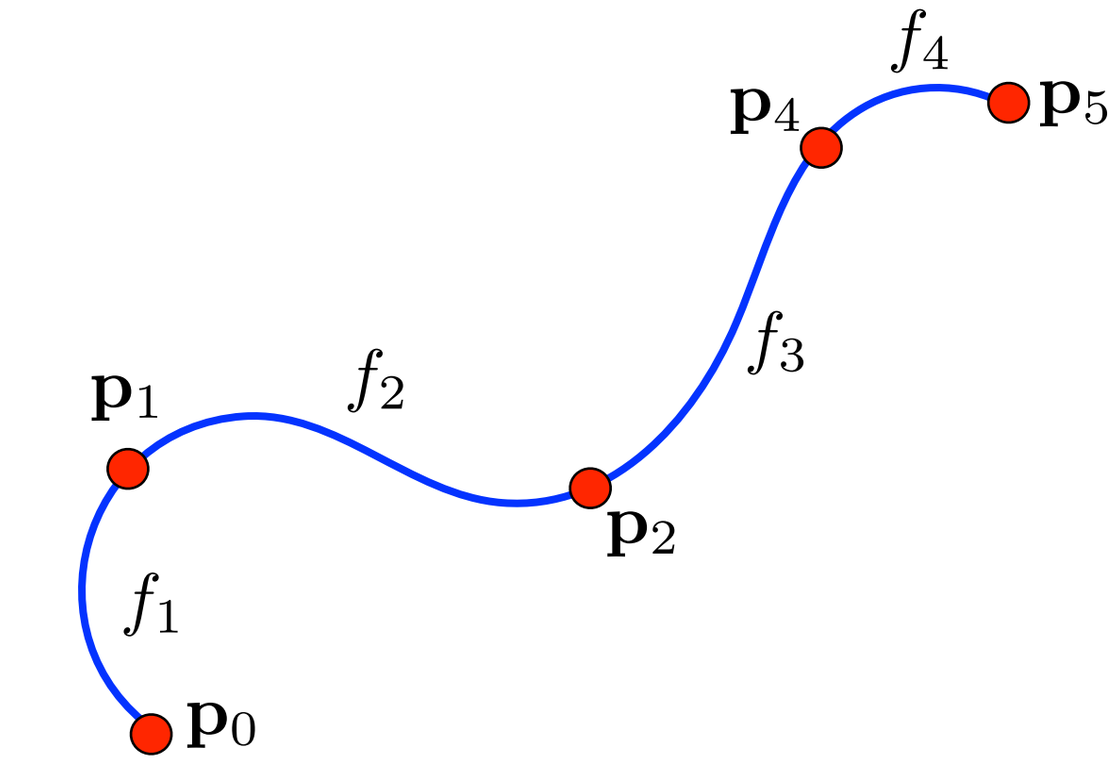


Interpolated and Approximated

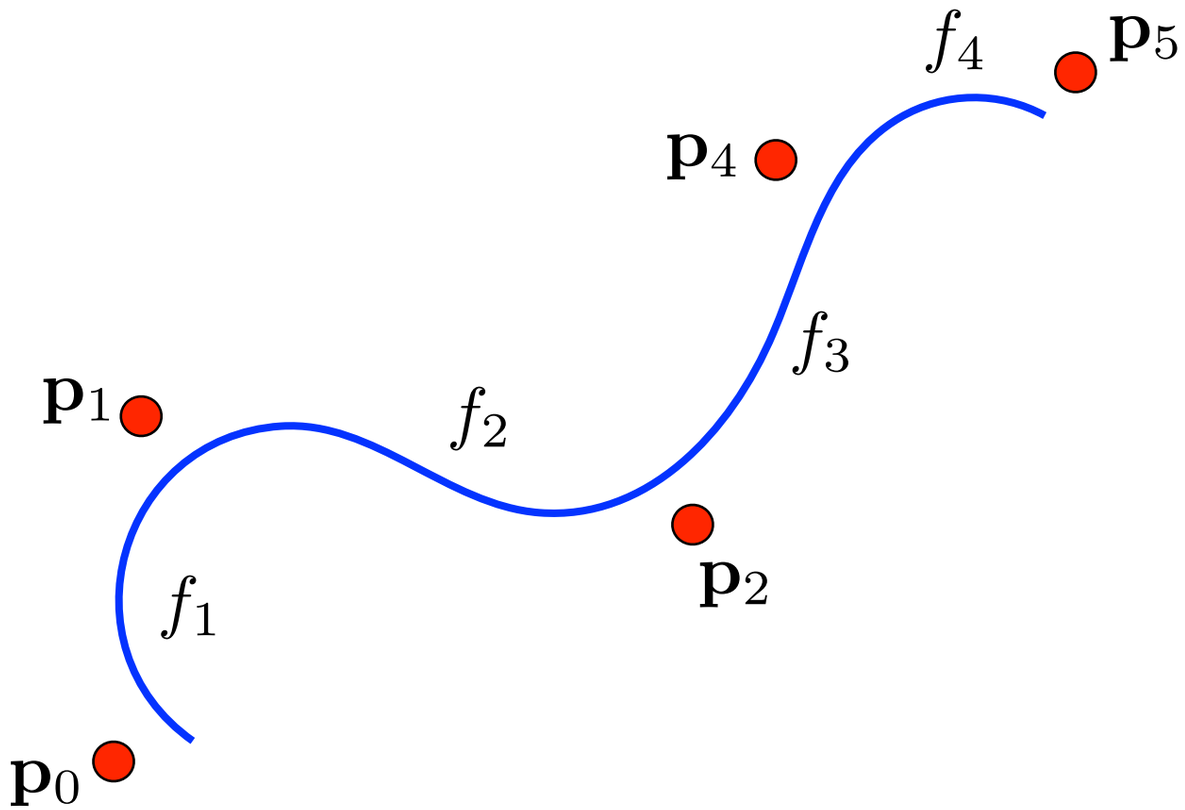
We specify a spline using **control points**

If the resulting curve fits with the control points, we talk about an **interpolated** curve.

If the control points are not part of the curve but determine the general direction, we talk about an **approximated** curve.



interpolated

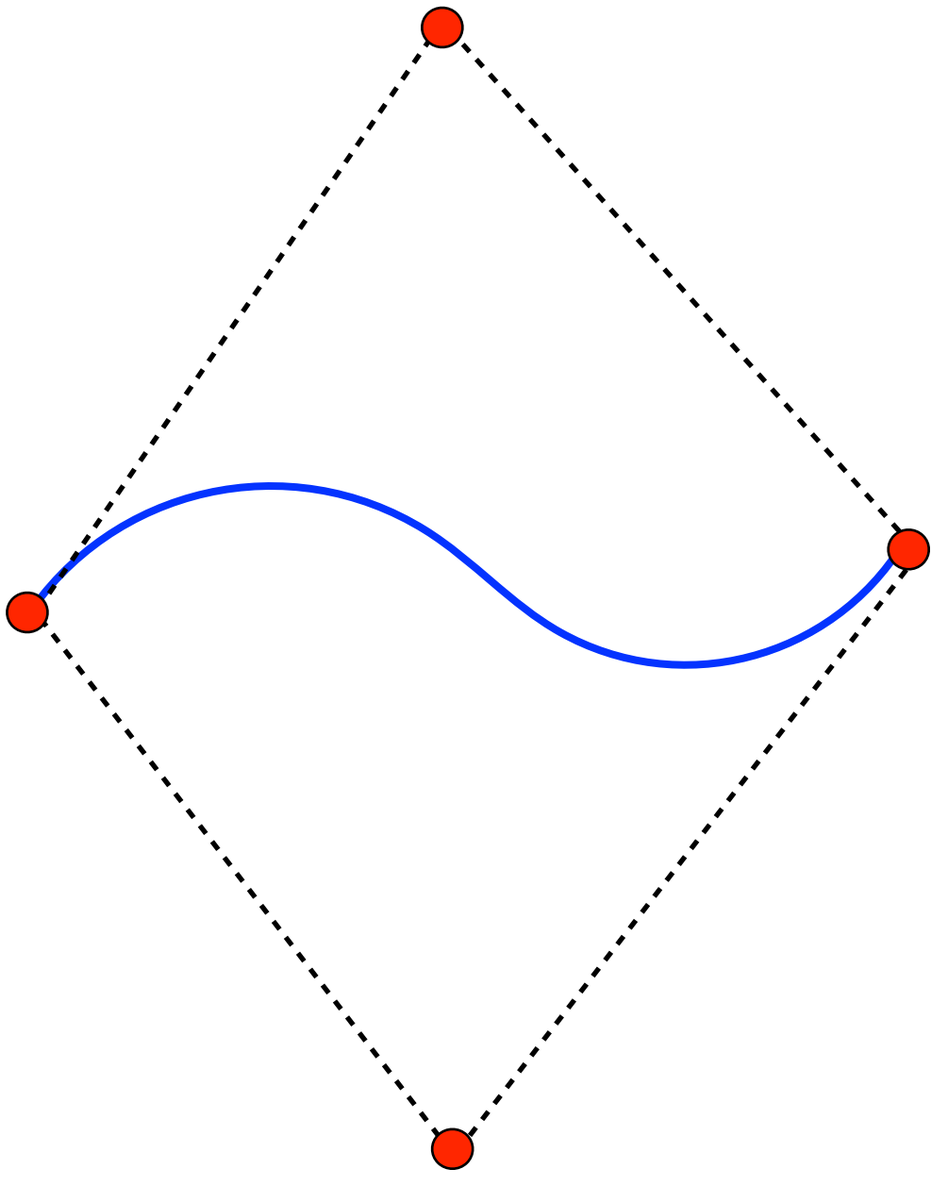


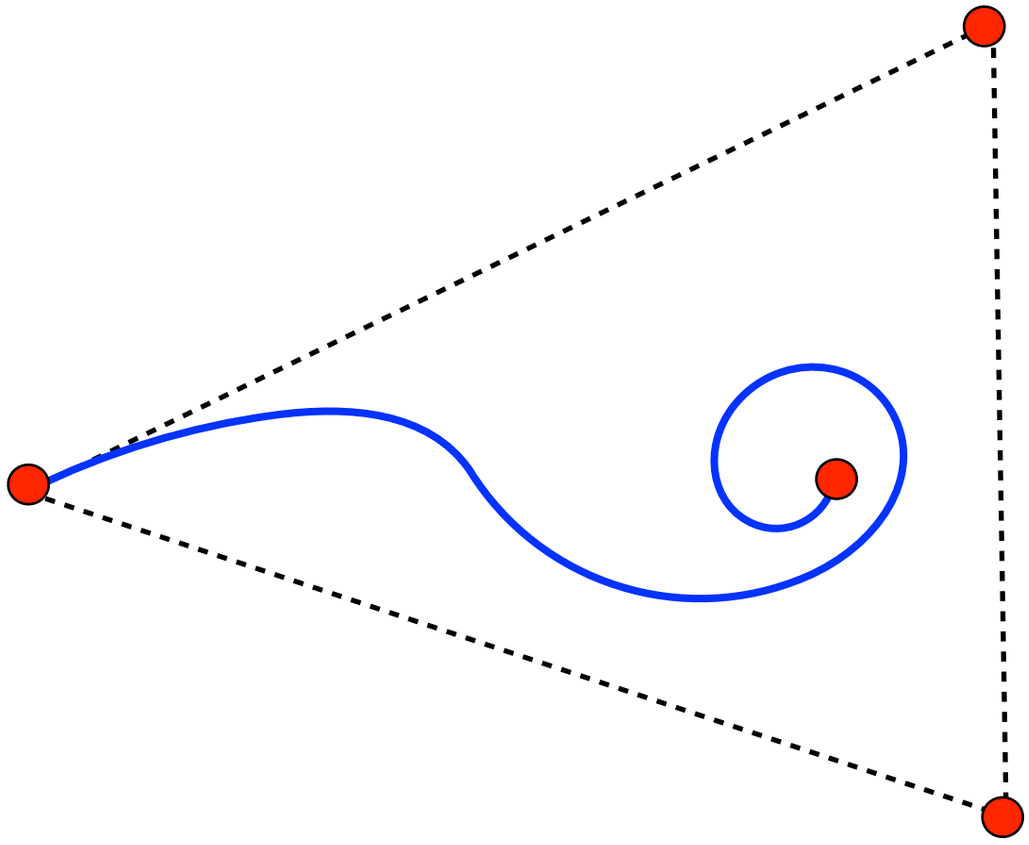
approximated

f_1 :function

Convex Hull

If a set of control points forms the boundaries for a region in space, we call this a **convex hull.**





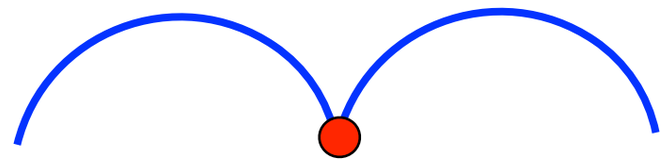
Parametric Continuity Conditions

We must ensure a smooth transition between the different spline functions.

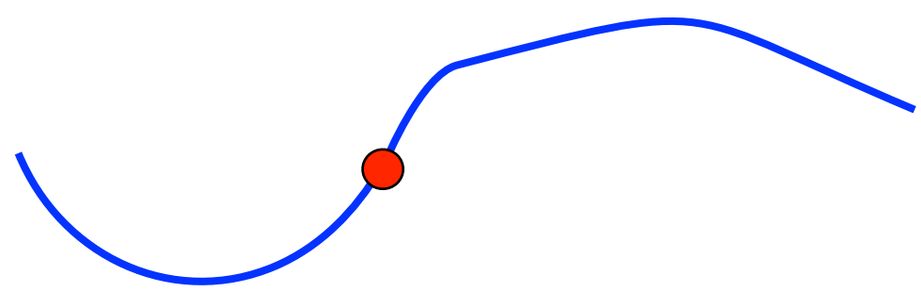
Therefore, we impose **continuity conditions**. Let's presume each section of the spline is represented by a set of parametric condition of the form

$$x = x(u), \quad y = y(u), \quad z = z(u), \quad u_1 \leq u \leq u_2$$

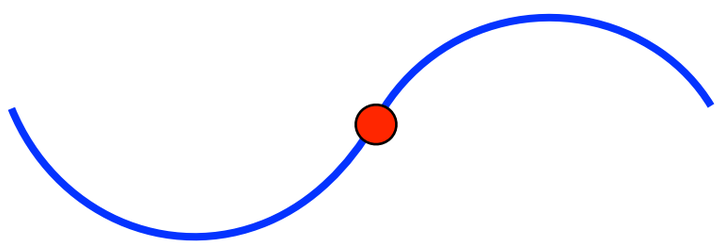
Zero-order parametric continuity C^0
means, the curves meet



First-order parametric continuity C^1
means, the first derivative (tangent lines) are equal at the meeting point



Second-order parametric continuity C^2
means, the first and second derivative are equal at the meeting point



Geometric Continuity Conditions

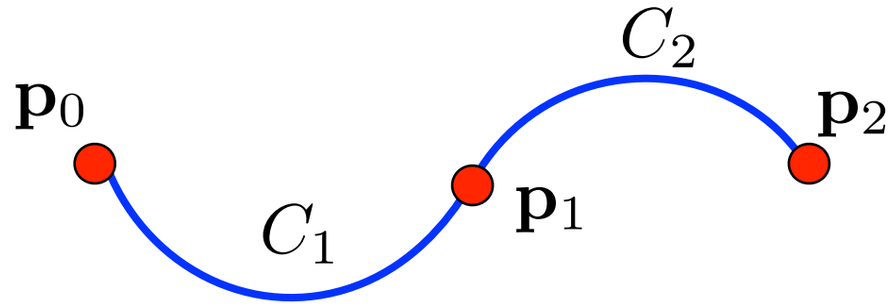
Geometric continuity conditions require that the parametric derivates are proportional to each other (instead of equality).

Zero-order geometric continuity G^0
means, the curves meet

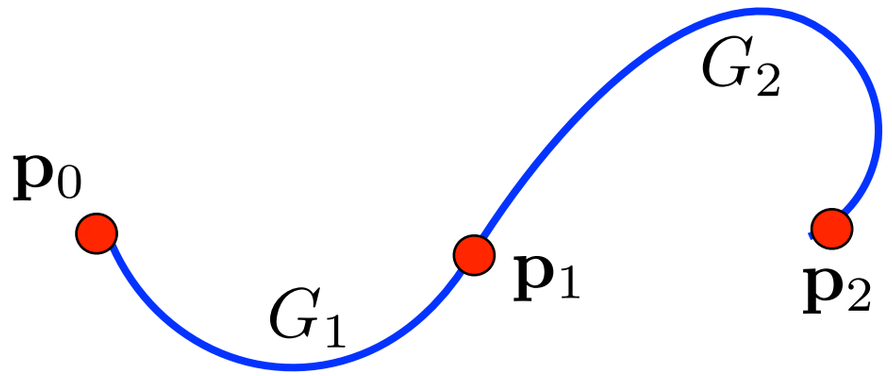
First-order geometric continuity G^1
means, the first derivative (tangent lines) are proportional at the intersection

Second-order geometric continuity G^2
means, the first and second derivative are proportional at the intersection point

The magnitude of all derivates can vary.



Parametric continuity



Geometric continuity

Spline Specification

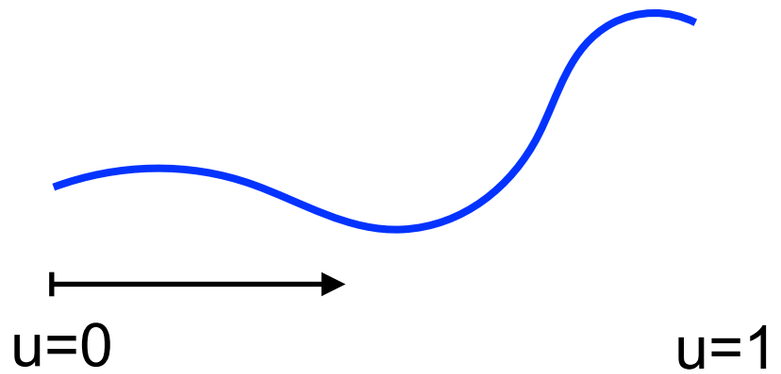
Let's start with a polynomial representation of a curve

$$x(u) = a_x\ u^3 + b_x\ u^2 + c_x\ u + d_x, \quad 0 \le u \le 1$$

Rewritten in matrix form.

$$x(u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \begin{bmatrix} a_x \\ b_x \\ c_x \\ d_x \end{bmatrix}$$

$$= \mathbf{U} \cdot \mathbf{C}$$



We always run from u = [0,1]

Let's split C into

$$\mathbf{C} = \mathbf{M}_{spline} \cdot \mathbf{M}_{geom}$$

where $\mathbf{M}_{geom}$ is a four-element column matrix containing the geometric constraints (boundary conditions).

and $\mathbf{M}_{spline}$ is a 4x4 matrix that transforms the geometric constraints into the polynomial coefficients a_x, b_x, c_x, d_x.

Spline Specification

Thus, we finally represent a spline as

$$x(u) = \mathbf{U} \cdot \mathbf{M}_{spline} \cdot \mathbf{M}_{geom}$$

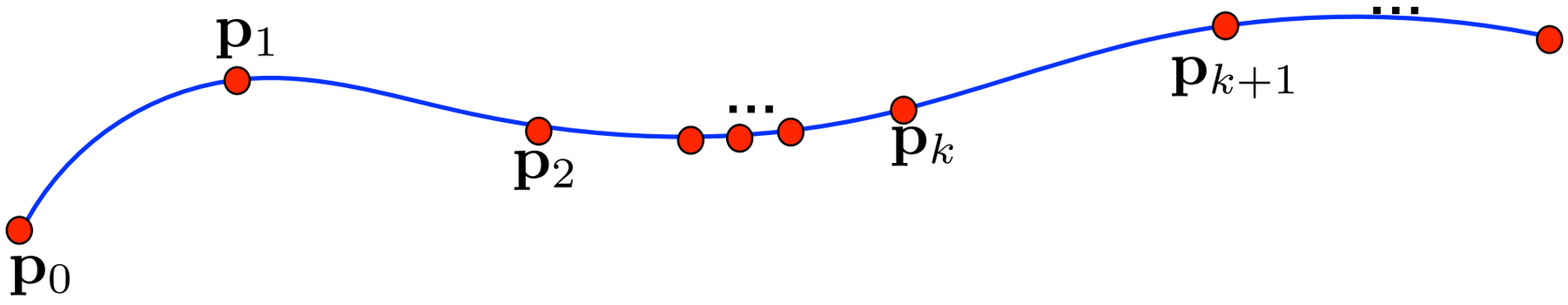
The matrix $\mathbf{M}_{spline}$ characterizes the spline representation. It is often called the **basis matrix**.

Cubic Spline Interpolation Methods

Given a set of control points, cubic splines are obtained by fitting the input points into with a piecewise cubic polynomial curve that passes through every control point.

With n curve sections, suppose we have n+1 points with the coordinates

$$\mathbf{p}_k = \{x_k, y_k, z_k\}, \quad k = 0, 1, 2, ..., n$$



We can describe a parametric curve that fits between each pair of control points with the following equation

$$x(u) = a_x\, u^3 + b_x\, u^3 + c_x\, u + d_x$$
$$y(u) = a_y\, u^3 + b_y\, u^3 + c_y\, u + d_y \qquad 0 \le u \le 1$$
$$z(u) = a_z\, u^3 + b_z\, u^3 + c_z\, u + d_z$$

Cubic Spline Interpolation Methods

- For each of these three equations, we need to compute the values for the coefficients *a, b, c, d*.
- We do this by setting enough boundary conditions.

Advantages

of cubic splines in comparison to quadratic polynomial splines

- flexible
- fast computation
- less memory
- more stable

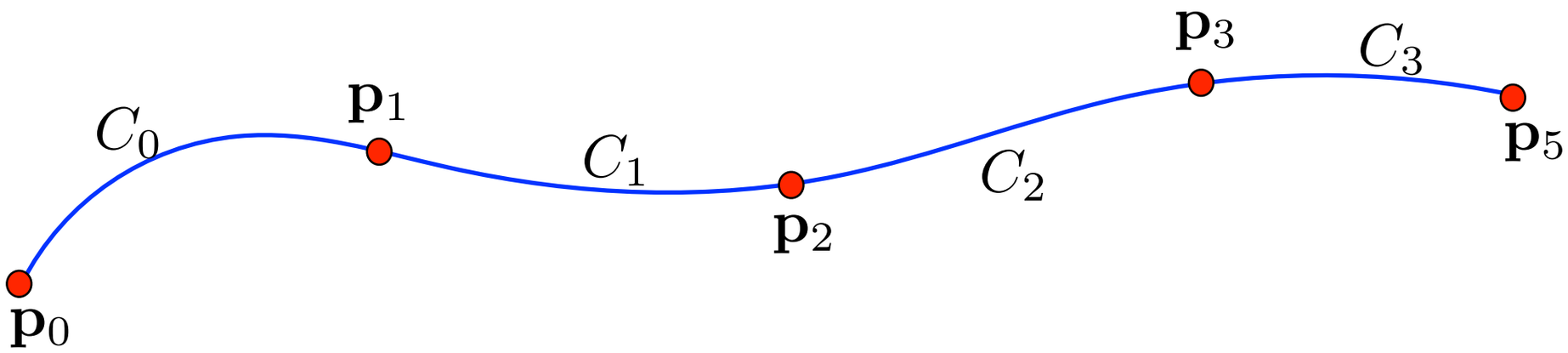
Different Methods

- Natural Cubic Spline
- Hermite Spline
- Cardinal Splines
- Kochanek-Bartels Splines

Natural Cubic Spline

It was the first, but obsolete today.

- Mathematic representation of the original draft spline
- Suppose we have <u>four</u> spline sections n, and n+1 control points



- This results in 4n polynomial coefficients *a, b, c*, and *d*.
- We have four boundary conditions. Each intersecting curve at each control point must have the same boundary condition.
- This gives us 3 equations + 1 equation for the start point.
- Result in 4*n* - 4 equations.
- <u>Disadvantage</u>: change of one condition or point affects the entire system of equations.

Hermite Spline

A Hermite (after the French mathematician Charles Hermite) spline is a piecewise cubic polynomial function with a specified tangent at each control point.

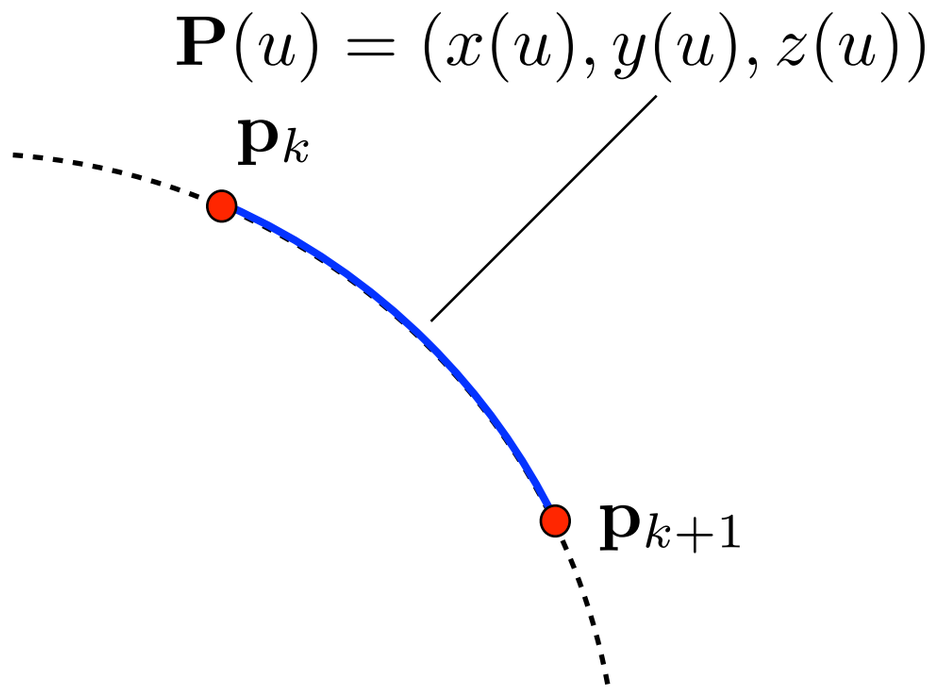
The spline can be adjust locally because it only depends on its endpoint constraints.

$\mathbf{P}(u)$ represents a parametric cubic point function for the curve section between the points p_k and p_{k+1}

Then, the boundary conditions are defined by:

$$\mathbf{P}(0) = \mathbf{p}_k$$
$$\mathbf{P}(1) = \mathbf{p}_{k+1}$$
$$\mathbf{P}'(0) = \mathbf{D}\ \mathbf{p}_k$$
$$\mathbf{P}'(1) = \mathbf{D}\ \mathbf{p}_{k+1}$$

Note, we need to specify these conditions = points + endpoint slopes

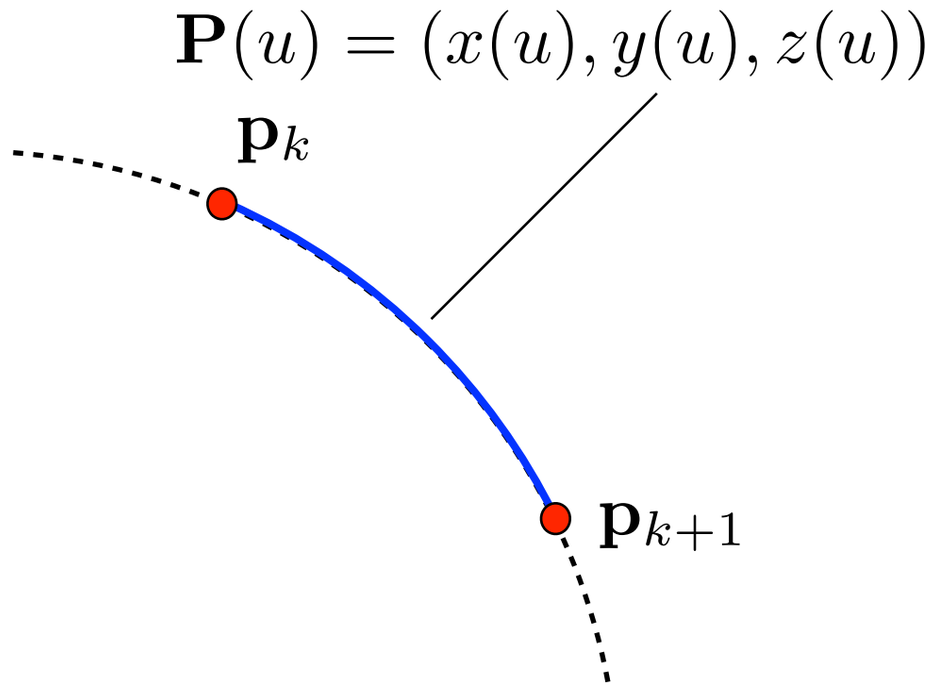


with $\mathbf{D}\ \mathbf{p}_k$, $\mathbf{D}\ \mathbf{p}_{k+1}$ the first derivative, the slope of the curve.

Hermite Spline

Add the polynomial curve

$$\mathbf{P}(u) = \mathbf{a}\,u^3 + \mathbf{b}\,u^2 + \mathbf{c}\,u + \mathbf{d}, \quad 0 \le u \le 1$$



note, **a**, **b**, **c**, and **d** in vector form to represent the components for x, y, z

In matrix form

$$\mathbf{P}(u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{bmatrix}$$

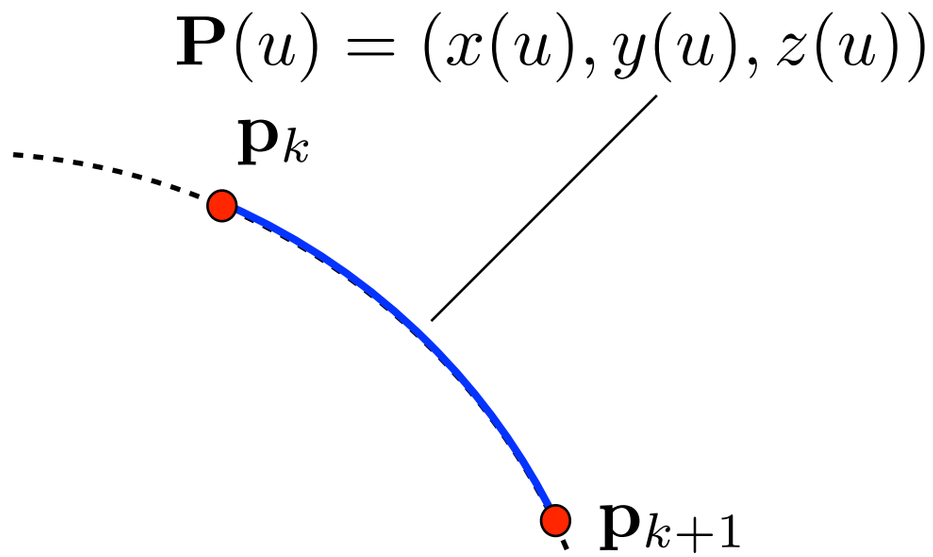
and the first derivative

$$\mathbf{P}'(u) = \begin{bmatrix} 3u^2 & 2u^1 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{bmatrix}$$

Hermite Spline

Substitute the endpoint values 0 and 1 for the parameter *u,* we can express the boundary conditions in matrix form.

$$\begin{bmatrix} \mathbf{p}_k \\ \mathbf{p}_{k+1} \\ \mathbf{D}\ \mathbf{p}_k \\ \mathbf{D}\ \mathbf{p}_{k+1} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{bmatrix}$$



solving for the parameters

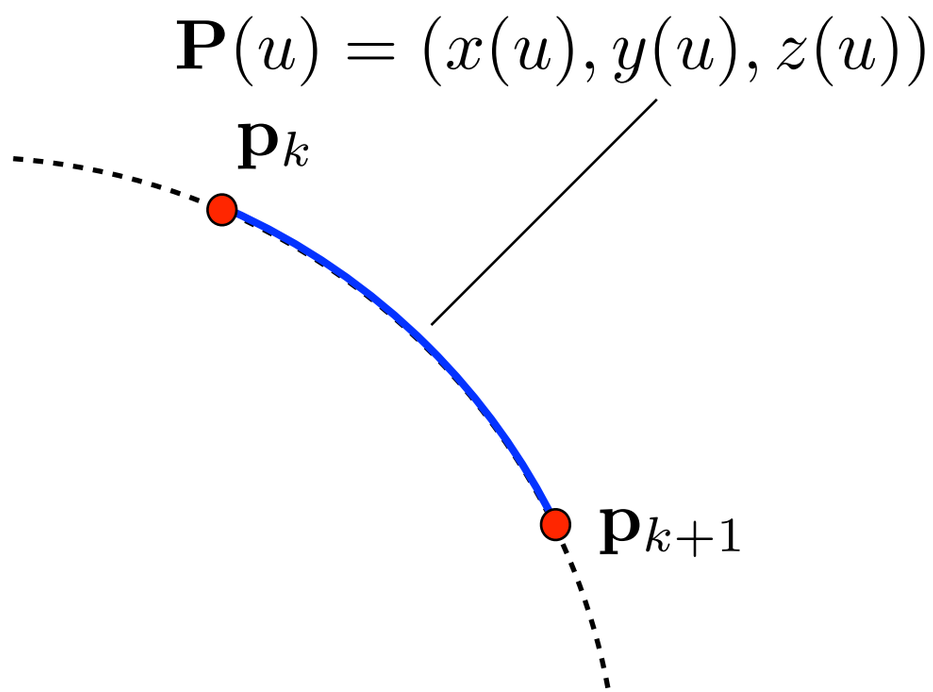
$$\begin{bmatrix} \mathbf{a} \\ \mathbf{b} \\ \mathbf{c} \\ \mathbf{d} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix}^{-1} \cdot \begin{bmatrix} \mathbf{p}_k \\ \mathbf{p}_{k+1} \\ \mathbf{D}\ \mathbf{p}_k \\ \mathbf{D}\ \mathbf{p}_{k+1} \end{bmatrix}$$

$$= \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{p}_k \\ \mathbf{p}_{k+1} \\ \mathbf{D}\ \mathbf{p}_k \\ \mathbf{D}\ \mathbf{p}_{k+1} \end{bmatrix}$$

Hermite Spline

$$= \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} \mathbf{p}_k \\ \mathbf{p}_{k+1} \\ \mathbf{D}\ \mathbf{p}_k \\ \mathbf{D}\ \mathbf{p}_{k+1} \end{bmatrix}$$

$$= \mathbf{M}_H \cdot \begin{bmatrix} \mathbf{p}_k \\ \mathbf{p}_{k+1} \\ \mathbf{D}\ \mathbf{p}_k \\ \mathbf{D}\ \mathbf{p}_{k+1} \end{bmatrix}$$



with **M**H, the **Hermite matrix**, which is the inverse of the boundary constraint matrix, thus can be rewritten as:

$$\mathbf{P}(u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \cdot \mathbf{M}_H \cdot \begin{bmatrix} \mathbf{p}_k \\ \mathbf{p}_{k+1} \\ \mathbf{D}\ \mathbf{p}_k \\ \mathbf{D}\ \mathbf{p}_{k+1} \end{bmatrix}$$

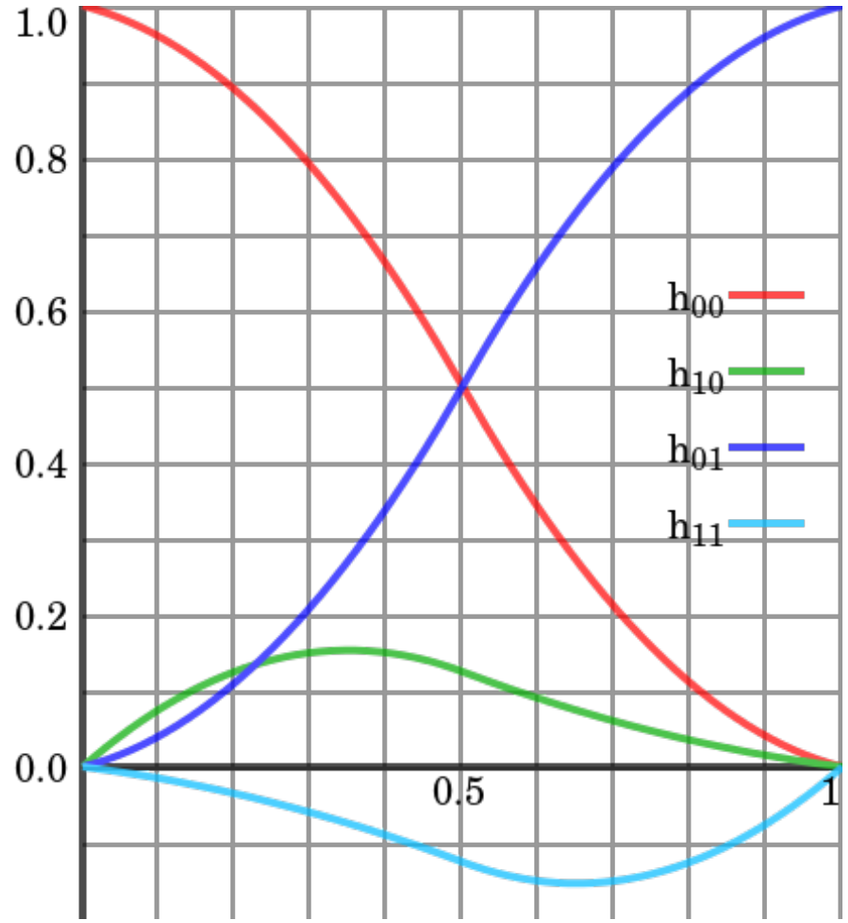
Hermite Spline

We can express this a polynomial function:

$$\begin{aligned}\mathbf{P}(u) =& \mathbf{p}_k(2u^3 - 3u^2 + 1) + \mathbf{p}_{k+1}(-2u^3 + 3u^2) + \\ & \mathbf{D}\ \mathbf{p}_k(u^3 - 2u^2 + u) + \mathbf{D}\ \mathbf{p}_{k+1}(u^3 - u^2) \\ =& \mathbf{p}_k H_0 + \mathbf{p}_{k+1} H_1 + \mathbf{D}\ \mathbf{p}_k H_2 + \mathbf{D}\ \mathbf{p}_{k+1} H_3\end{aligned}$$

with H_i, the **Hermite basis function**

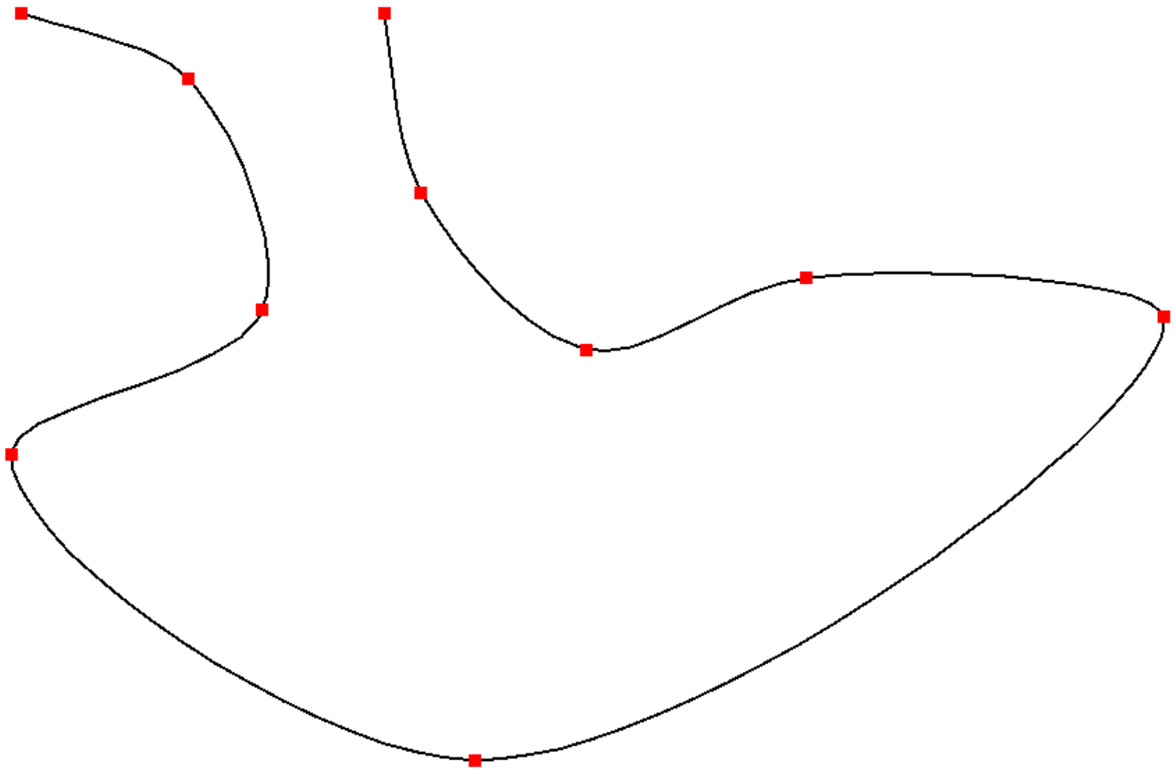
The result in each subinterval is a linear combination of these four functions.(u)

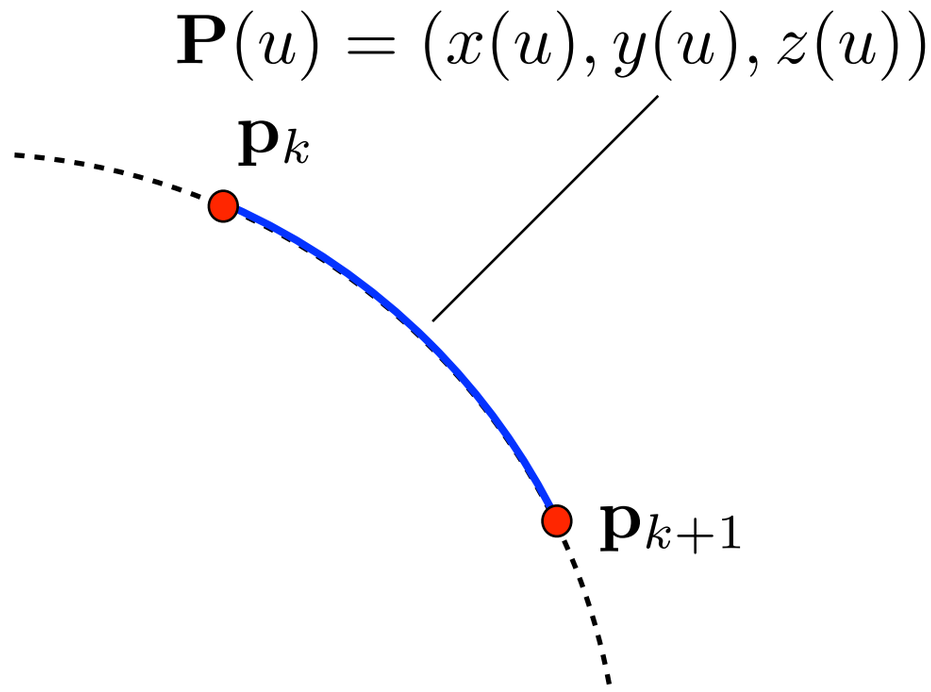


[wikipedia]

Hermite Spline

Useful for all curves where one can specify the curve slope



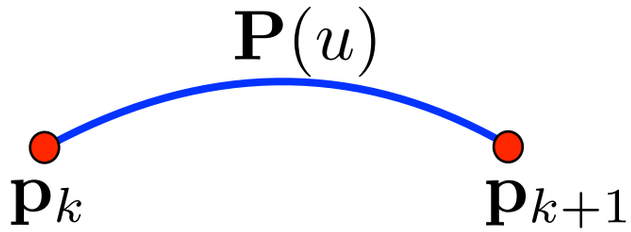


We need to specify the **parametric** conditions for all control points.

$$\mathbf{P}(0) = \mathbf{p}_k$$
$$\mathbf{P}(1) = \mathbf{p}_{k+1}$$
$$\mathbf{P}'(0) = \mathbf{D}\ \mathbf{p}_k$$
$$\mathbf{P}'(1) = \mathbf{D}\ \mathbf{p}_{k+1}$$

Cardinal Splines

Similar to the Hermite spline, the Cardinal Spline is a piecewise cubic polynomial function with a specified tangent at each control point. The difference is that we do not have to specify the endpoint tangents, the slope at the control point is calculated from the two adjacent control points.



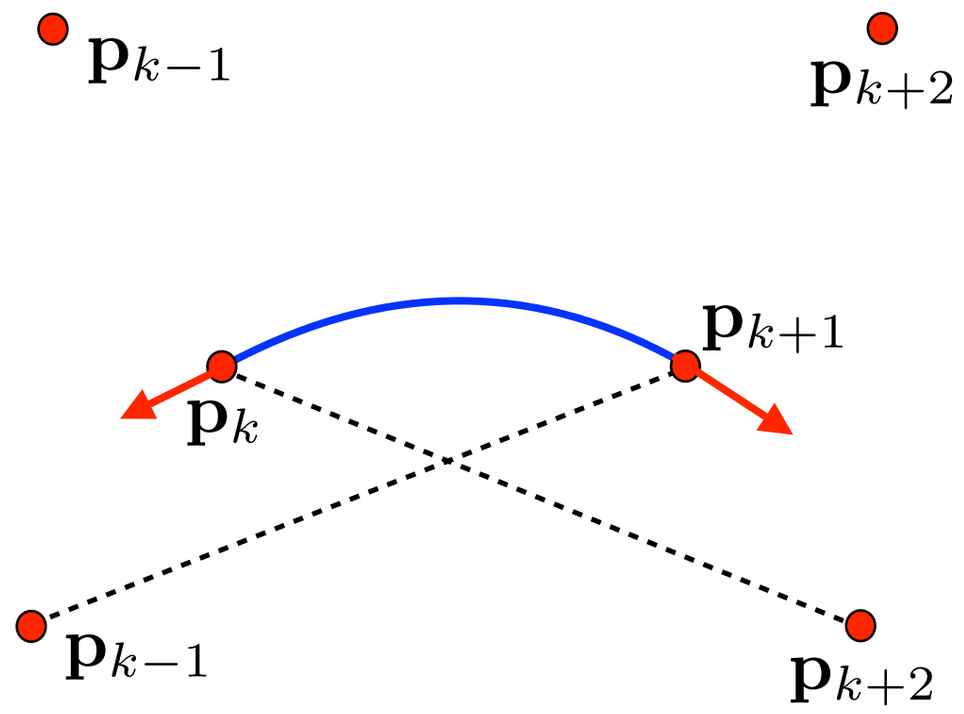
We use the boundary conditions

$$\mathbf{P}(0) = \mathbf{p}_k$$

$$\mathbf{P}(1) = \mathbf{p}_{k+1}$$

$$\mathbf{P}'(0) = \frac{1}{2}(1-t)(\mathbf{p}_{k+1} - \mathbf{p}_{k-1})$$

$$\mathbf{P}'(1) = \frac{1}{2}(1-t)(\mathbf{p}_{k+2} - \mathbf{p}_k)$$

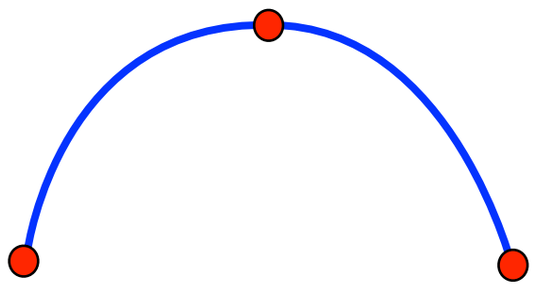


Thus, the slopes at the control points are proportional to the chords

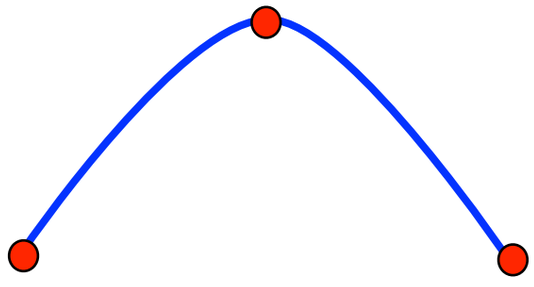
$\overline{\mathbf{p}_{k+1}\ \mathbf{p}_{k-1}}$ and $\overline{\mathbf{p}_k\ \mathbf{p}_{k+2}}$

Cardinal Splines

The parameter *t* is called the tension parameter: tight corners to round corners



t < 0



t > 0

with t = 0: Catmull-Rom splines

Cardinal Splines

Starting with the equation

$$\mathbf{P}(u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \cdot \mathbf{M}_C \cdot \begin{bmatrix} \mathbf{p}_{k-1} \\ \mathbf{p}_k \\ \mathbf{p}_{k+1} \\ \mathbf{p}_{k+2} \end{bmatrix}$$

the Cardinal matrix $\mathbf{M}_C$ is:

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

with $s = (1-t)/2$

Cardinal Splines

Rewritten in polynomial form

$$\begin{aligned}\mathbf{P}(u) =& \mathbf{p}_{k-1}(-s\ u^3 + 2s\ u^2 - s) + \mathbf{p}_k[(2-s)u^3 + (s-3)u^2 + 1] + \\ & \mathbf{p}_{k+1}[(s-2)u^3 + (3-2s)u^2 + s\ u] + \mathbf{p}_{k+2}(s\ u^3 - s\ u^2) \\ =& \mathbf{p}_{k-1}\mathrm{CAR}_0(u) + \mathbf{p}_k\mathrm{CAR}_1(u)\mathbf{p}_{k+1}\mathrm{CAR}_2(u) + \mathbf{p}_{k+2}\mathrm{CAR}_3(u)\end{aligned}$$

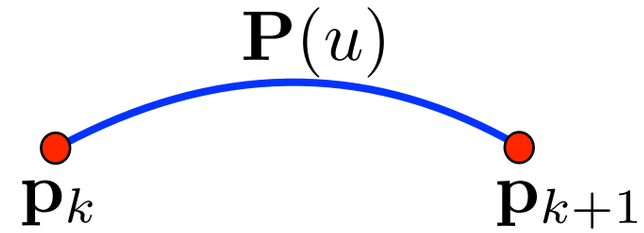
with the Cardinal spline function CAR_k

Kochanek-Bartels Splines

This is an extension of the Cardinal spline.

Two additional parameters are introduced to allow more shape flexibility.

Given the four control points



$$\mathbf{P}(0) = \mathbf{p}_k$$

$$\mathbf{P}(1) = \mathbf{p}_{k+1}$$

$$\mathbf{P}'(0)_{in} = \frac{1}{2}(1-t)\left[(1+b)(1-c)(\mathbf{p}_k - \mathbf{p}_{k-1})\right.$$
$$\left. + (1-b)(1+c)(\mathbf{p}_{k+1} - \mathbf{p}_k)\right.$$

$$\mathbf{P}'(1)_{in} = \frac{1}{2}(1-t)\left[(1+b)(1+c)(\mathbf{p}_{k+1} - \mathbf{p}_k)\right.$$
$$\left. + (1-b)(1-c)(\mathbf{p}_{k+2} - \mathbf{p}_{k+1})\right.$$

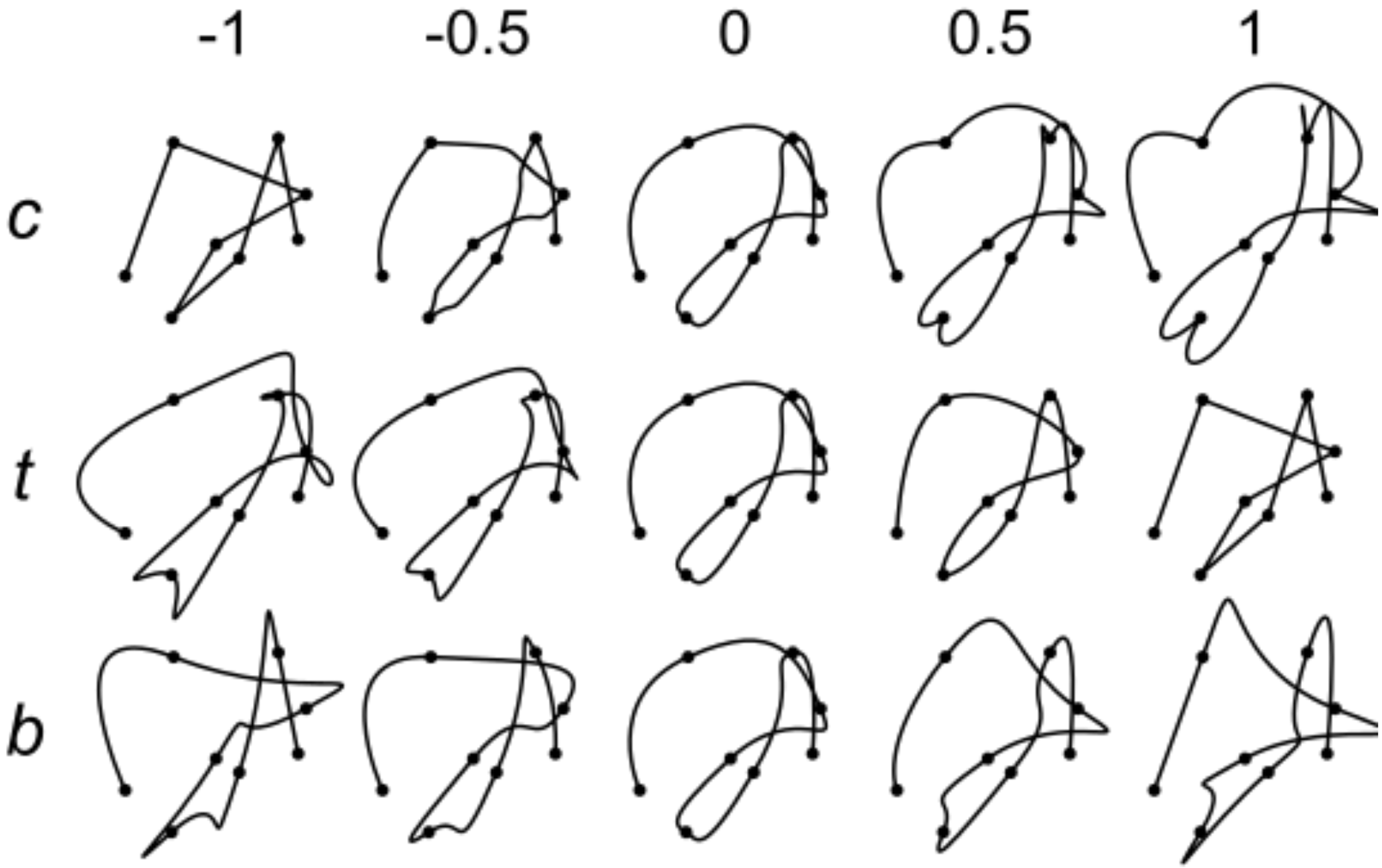
Kochanek-Bartels Splines

ARLAB

Two new parameters:

b: bias

c: continuity



[wikipedia]

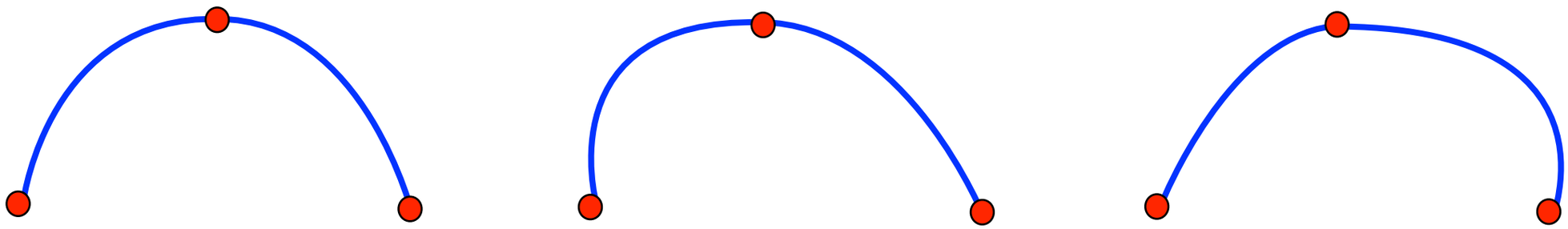
VRAC|HCI

IOWA STATE UNIVERSITY
OF SCIENCE AND TECHNOLOGY

Kochanek-Bartels Splines

Continuity: round corners to box corners

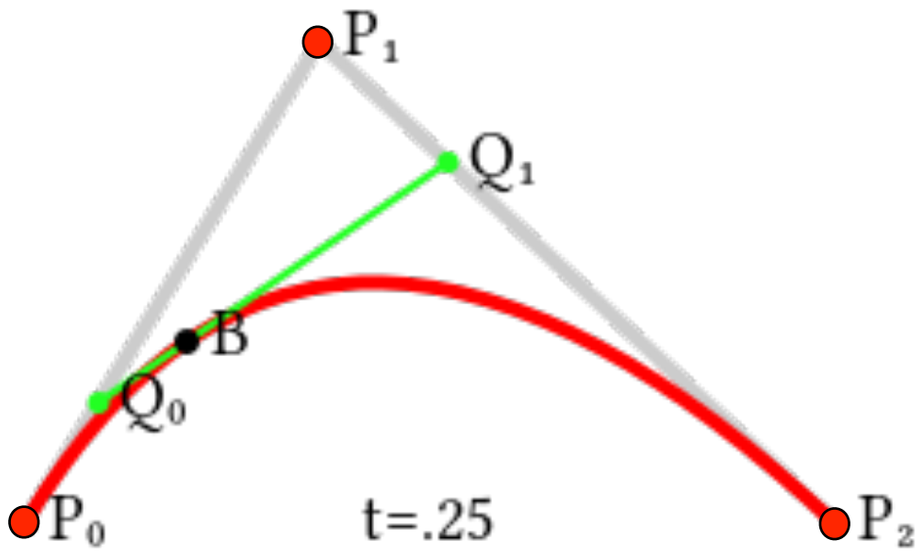
Bias: post - pre magnitude

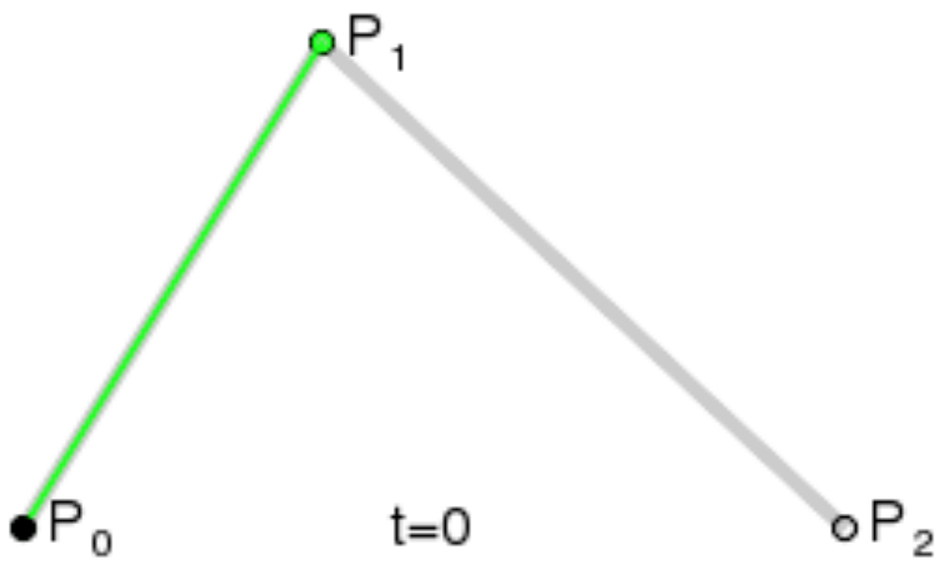


Bezier Spline Curves

Bezier splines was developed by the French engineer Pierre Bezier for the car design at Renault.

- They can be fitted to any number of control points.
- Bridge computer graphics and computer-aided design
- Can be modified with blending functions, approximated, etc.
- They are very easy to implement.





Quadratic curves, [wikipedia]

Bezier Spline Equation

We start with the general equation for Bezier Splines of every order!

Given a set of control points for *n* curve sections, with *n*+1 control points

$$\mathbf{p}_k = \{x_k, y_k, z_k\}, \quad k = 0, 1, 2, ..., n$$

k, index of the point
n, number of points

We define a position vector to draw a line as

$$\mathbf{P}(u) = \sum_{k=0}^{n} \mathbf{p}_k \mathrm{BEZ}_{k,n}(u), \quad 0 \leq u \leq 1$$

It describes the path between $\mathbf{p}_0$ and $\mathbf{p}_1$

The **Bezier blending function** is the *Bernstein polynomials*

$$\mathrm{BEZ}_{k,n}(u) = C(n,k)u^k(1-u)^{n-k}$$

with the parameter C:

$$C(n,k) = \frac{n!}{k!(n-k)!}$$

Bezier Spline Equation

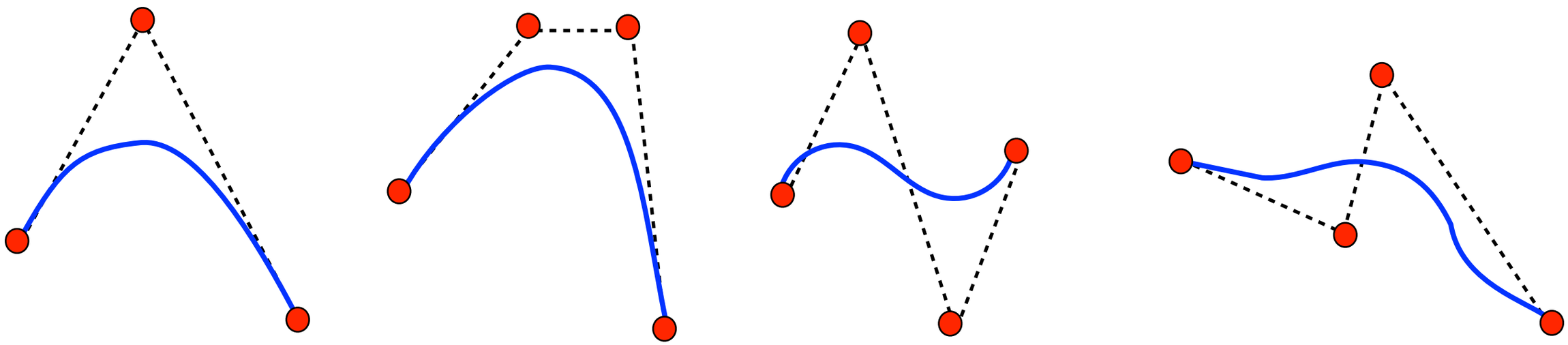
The vector **P**(u) represents the three parametric equations:

$$x(u) = \sum_{k=0}^{n} x_k \ \mathrm{BEZ}_{k,n}(u)$$

$$y(u) = \sum_{k=0}^{n} y_k \ \mathrm{BEZ}_{k,n}(u)$$

$$z(u) = \sum_{k=0}^{n} z_k \ \mathrm{BEZ}_{k,n}(u)$$

Bezier Spline Equation



The order of the Bezier curve is n-1 (n, number of control points)

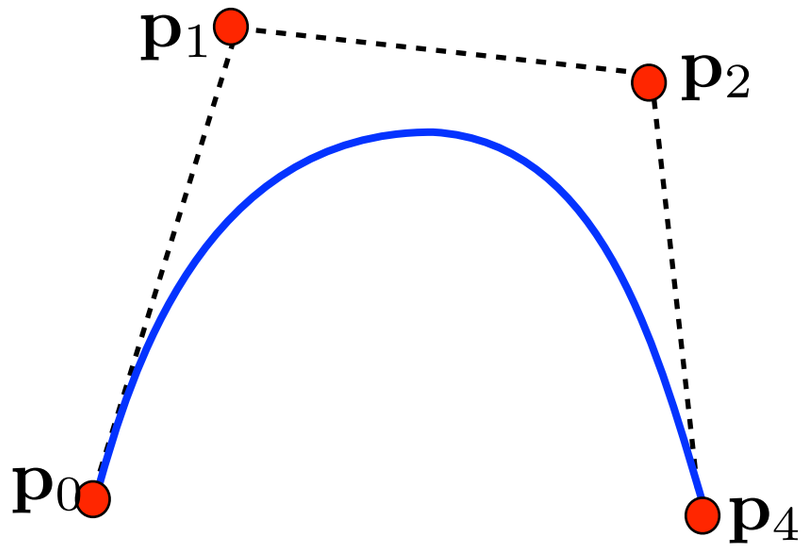
- quadratic or 2nd order curve: three points generate a parabola
- cubic or 3rd order curve: four points a cubic curve

etc.

Cubic Bezier Curve

For a Cubic Bezier Function, we use the following blending functions

$$\mathrm{BEZ}_{0,3}(u) = (1-u)^3$$
$$\mathrm{BEZ}_{1,3}(u) = 3u(1-u)^2$$
$$\mathrm{BEZ}_{2,3}(u) = 3u^2(1-u)$$
$$\mathrm{BEZ}_{3,3}(u) = u^3$$



The blending function determine how the control points influence the shape.

for u = 0: only the first blending function has an impact. Thus, we always begin with $\mathbf{p}_0$

for u = 1: only the last blending function has an impact, Thus, we will always end with $\mathbf{p}_3$

Each of this function is non zero in the range between 0 < u < 1

This also means, we have no local control points. The entire curve is always affected.

Cubic Bezier Curve

Back to the basic function:

$$\mathbf{P}(u) = \sum_{k=0}^{n} \mathbf{p}_k \mathrm{BEZ}_{k,n}(u), \quad 0 \leq u \leq 1$$

We need to follow the boundary conditions:

$$\mathbf{P}(0) = \mathbf{p}_0$$

$$\mathbf{P}(1) = \mathbf{p}_n$$

Along with the hire order boundary conditions

$$\mathbf{P}'(0) = -n\,\mathbf{p}_0 + n\,\mathbf{p}_1$$

$$\mathbf{P}'(1) = -n\,\mathbf{p}_{n-1} + n\,\mathbf{p}_n$$

$$\mathbf{P}''(0) = n(n-1)[(\mathbf{p}_2 - \mathbf{p}_1) - (\mathbf{p}_1 - \mathbf{p}_0)]$$

$$\mathbf{P}''(1) = n(n-1)[(\mathbf{p}_{n-2} - \mathbf{p}_{n-1}) - (\mathbf{p}_{n-1} - \mathbf{p}_n)]$$

Cubic Bezier Curve

For our cubic curve at the endpoints

$$\begin{aligned}\mathbf{P}'(0) &= 3(\mathbf{p}_1 - \mathbf{p}_0)\\ \mathbf{P}'(1) &= 3(\mathbf{p}_3 + \mathbf{p}_2)\\ \mathbf{P}''(0) &= 6(\mathbf{p}_0 - 2\mathbf{p}_1\mathbf{p}_2)\\ \mathbf{P}''(1) &= 6(\mathbf{p}_1 - 2\mathbf{p}_2 + \mathbf{p}_3)\end{aligned}$$

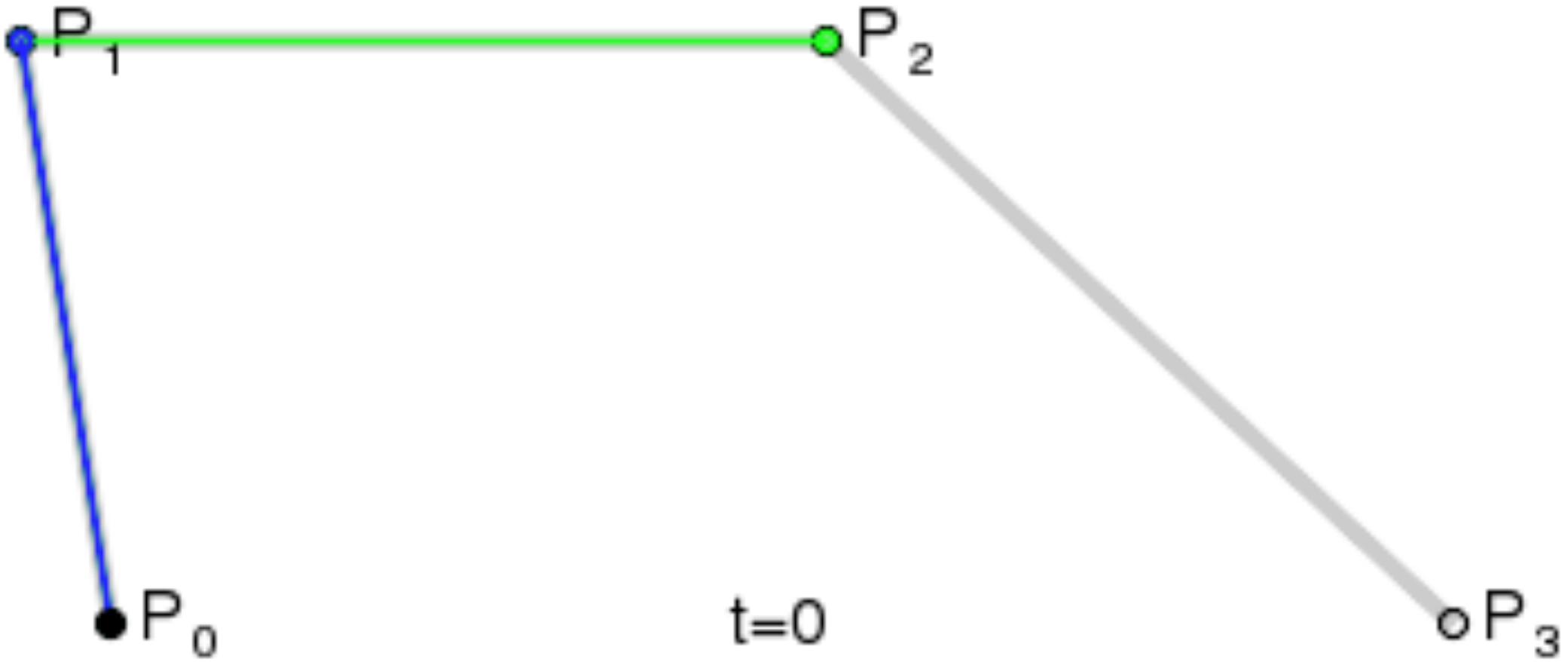
which we can transform into a matrix:

$$\mathbf{P}(u) = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \cdot \mathbf{M}_{Bez} \cdot \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix}$$

with the **Bezier matrix:**

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

Cubic Bezier Curve



Cubic curves, [wikipedia]

Thank you!

Questions

Rafael Radkowski, Ph.D.
Iowa State University
Virtual Reality Applications Center
1620 Howe Hall
Ames, Iowa 50011, USA

+1 515.294.5580
+1 515.294.5530(fax)

rafael@iastate.edu
http://arlab.me.iastate.edu

www.linkedin.com/in/rradkowski

ARLAB

VRAC|HCI

IOWA STATE UNIVERSITY
OF SCIENCE AND TECHNOLOGY