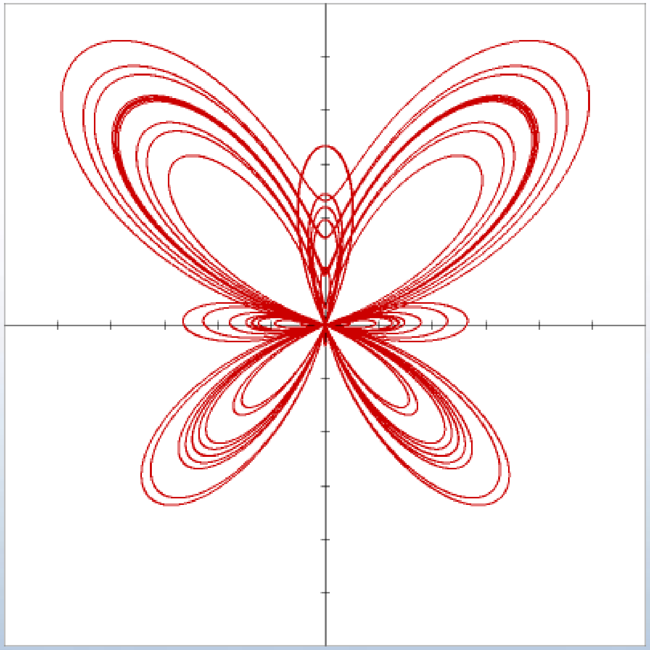
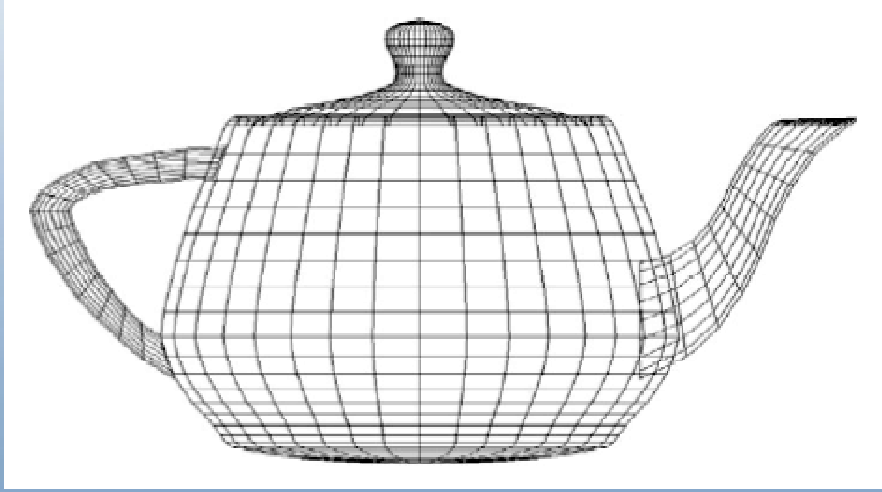


Parametric Objects



http://hm.softalliance.net/

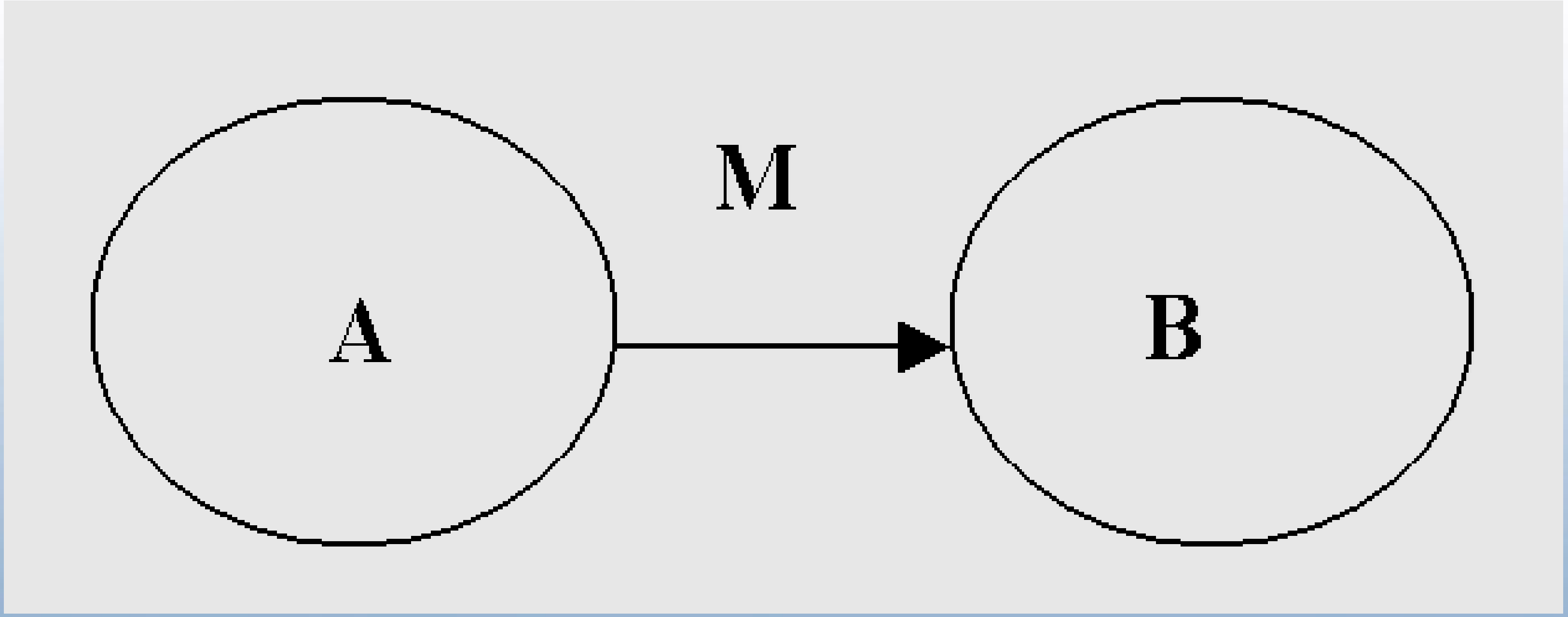
Alexander Pasko, Evgenii Maltsev

Contents

- *Parametric curves*
- *Polar coordinates*
- *Cylindrical coordinates*
- *Interpolation and approximation*
- *Parametric surfaces*
- *Spherical coordinates*
- *Trimmed parametric surfaces*

Mapping of a Known Set

$$M : A \rightarrow B$$



Parametric form

Parametric Curve Notion

A parametric curve is defined by a mapping of a unit segment to n-D space.

Parametric equations of a curve are obtained by introducing one more extra variable *t*, or a parameter, and calculating n-D point coordinates as functions of the parameter *t*:

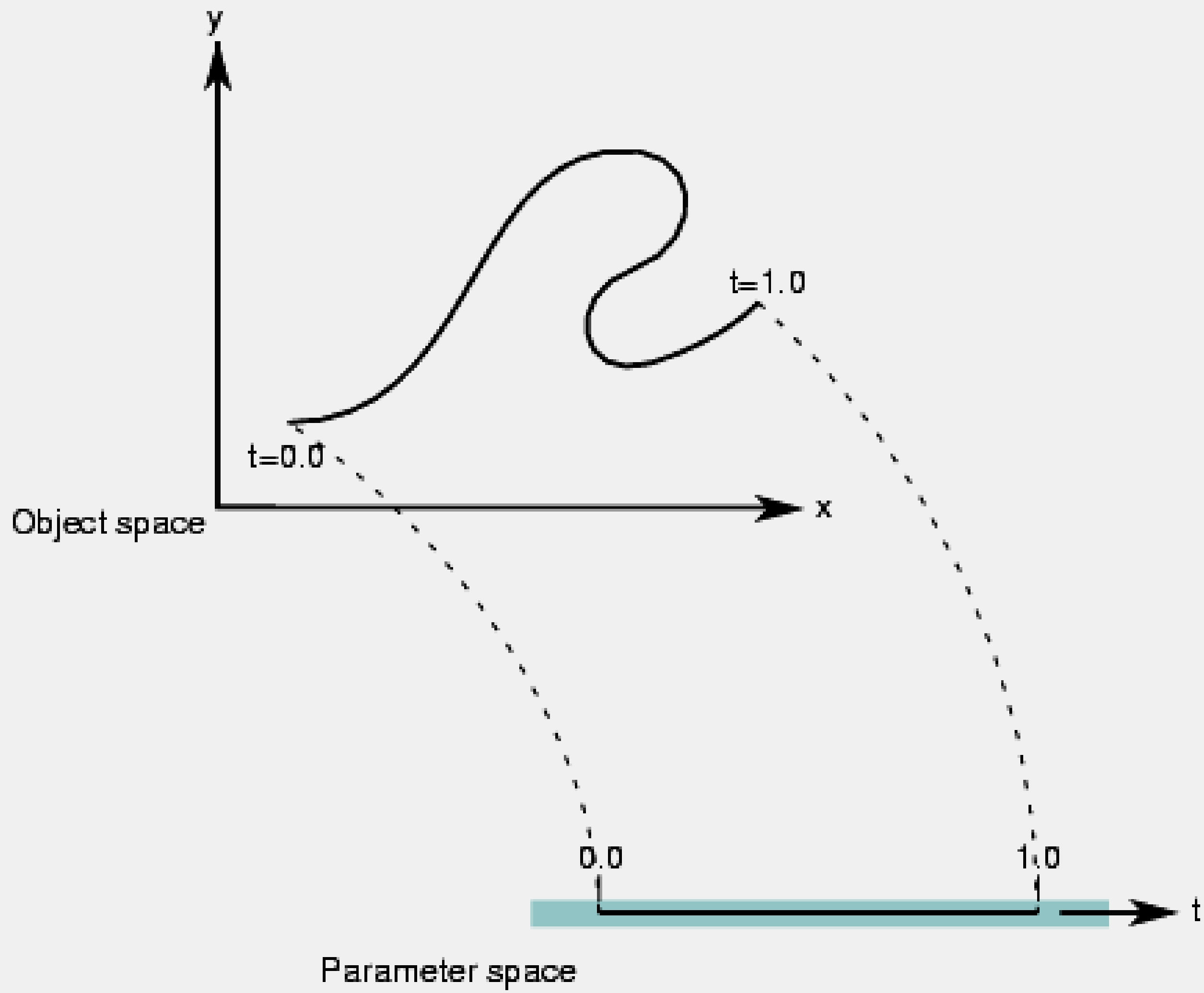
$$x_1 = \varphi_1(t)$$
$$x_2 = \varphi_2(t)$$
$$\ldots =$$
$$x_n = \varphi_n(t)$$

Parametric Curves

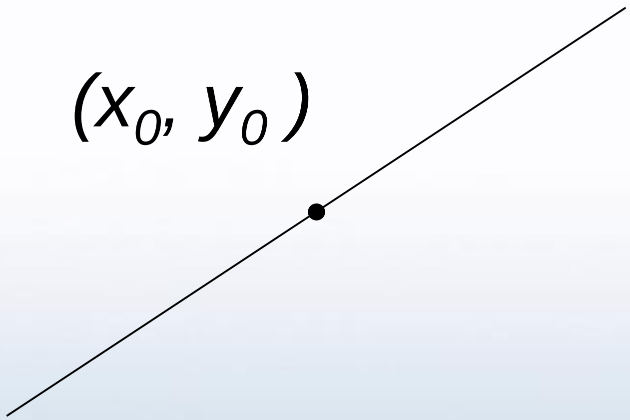
Planar curve
(2D space)

Each component of a point on the planar curve is a function of *t*, which lies in the parameter interval [0, 1] on the real line. Points on the curve are described by a pair of functions of *t*:

$x(t), y(t)$



Straight line and segment

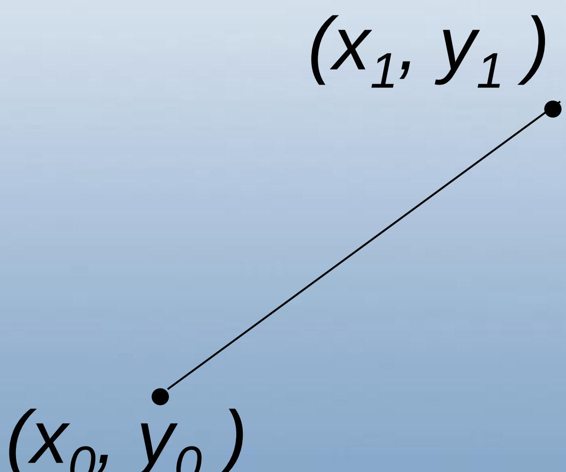


Infinite straight line

$x = x_0 + at$

$y = y_0 + bt$

$t \in]-\infty, +\infty[$



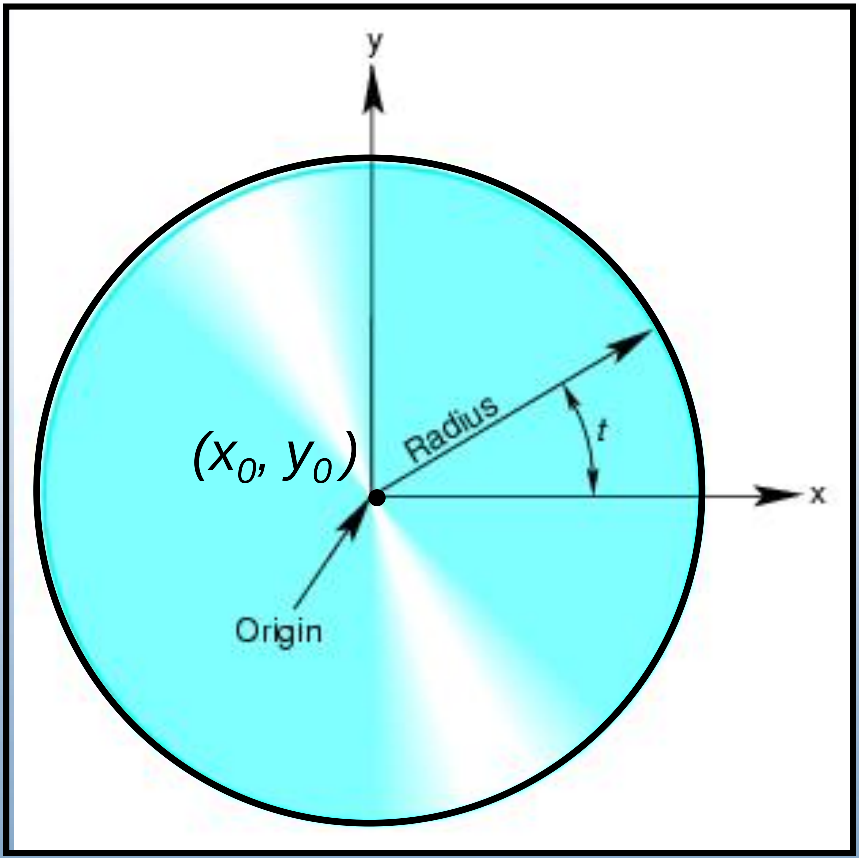
Straight line segment

$x = x_0 + (x_1 - x_0)t$

$y = y_0 + (y_1 - y_0)t$

$t \in [0, 1]$

Circle



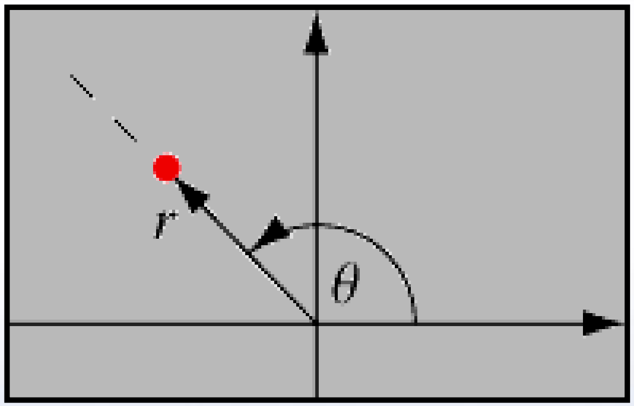
$$x = x_0 + R \cos t$$
$$y = y_0 + R \sin t$$

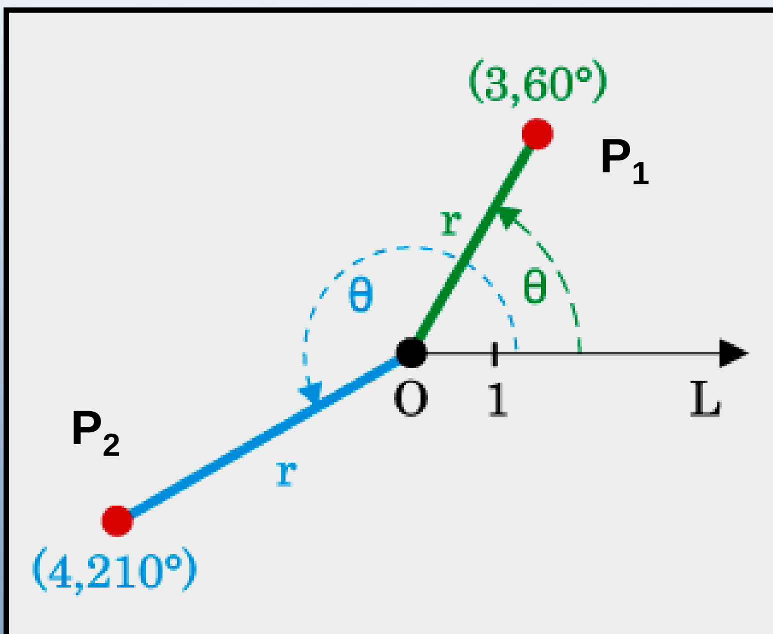
(x_0, y_0) circle center

R is a radius

$t \in [0, 2\pi]$ angle

Polar coordinate system





The polar coordinate system on a plane is defined by

- an origin, point *O*
- a semi-infinite line *L* leading from this point *(*polar axis)
- a point P representation by a tuple of two components (r, θ):

$r \geq 0$ is the distance from the origin to the point P

$0 \leq \theta \leq 360^o$ is the angle between the polar axis and the line from the origin to the point P.

Conversion between coordinate systems

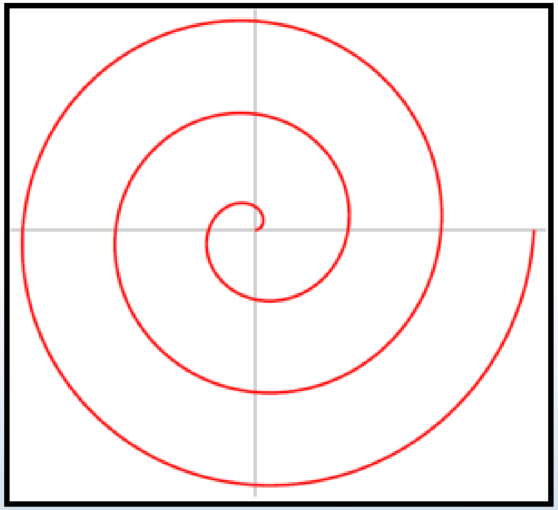
From polar to Cartesian coordinates:

$$x = r\cos\theta$$
$$y = r\sin\theta$$

From Cartesian to polar coordinates:

$$r = \sqrt{x^2 + y^2}$$
$$\theta = \arctan\frac{y}{x}$$

Spiral and Lissajous



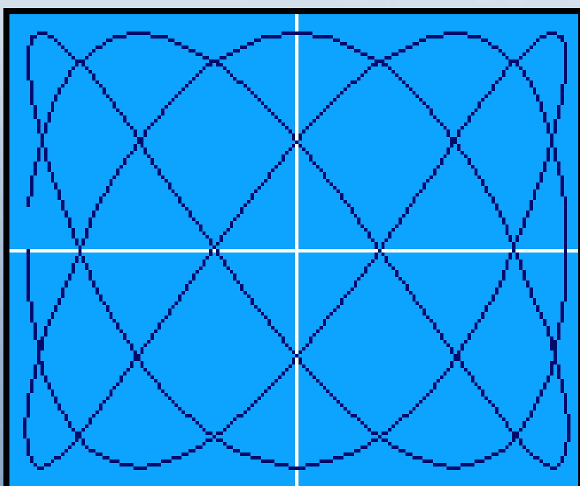
Archimedes spiral

Polar system:

$$r = t$$
$$\theta = t$$

Cartesian system:

$$x = t \cos t$$
$$y = t \sin t$$

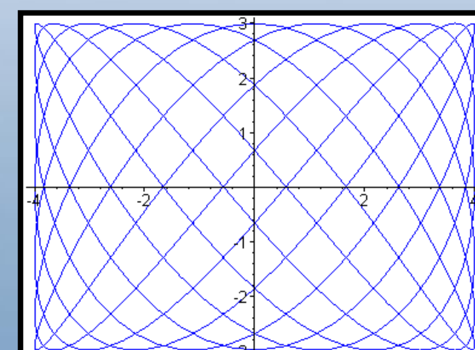


p = 3, q = 5

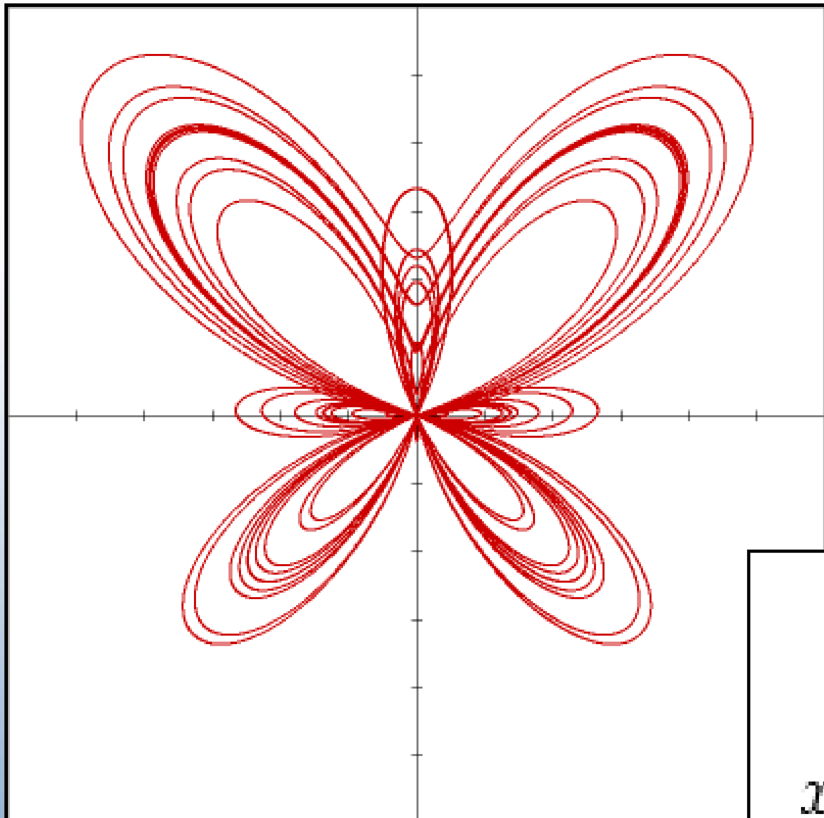
Lissajous curves

$$x = \cos pt$$
$$y = \sin qt$$

for any integer p, q



Butterfly curve



Discovered by Temple H. Fay

Polar system:

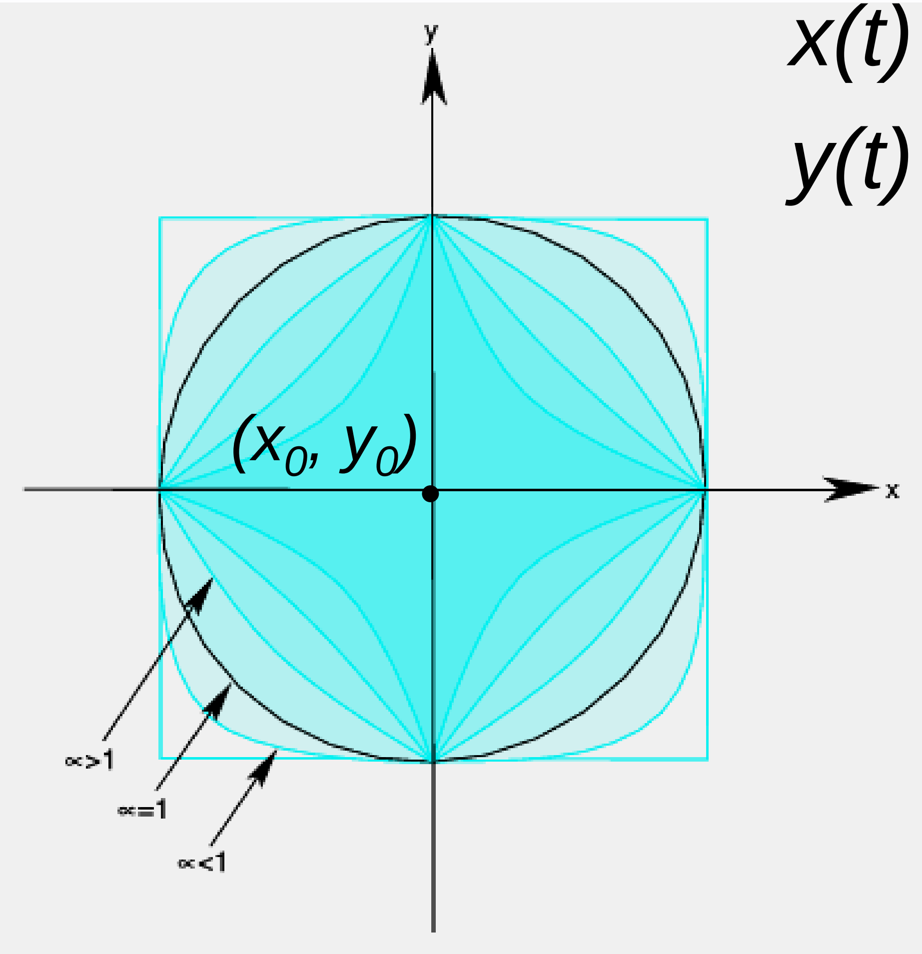
$$r = e^{sin\theta} - 2\cos(4\theta) + sin^5 \frac{1}{24}(2\theta - \pi)$$

Cartesian system:

$$x = \sin t \left[e^{\cos t} - 2\cos 4t - \sin^5 \frac{t}{12} \right]$$

$$y = \cos t \left[e^{\cos t} - 2\cos 4t - \sin^5 \frac{t}{12} \right]$$

Superquadric curves



$$x(t) = R\,(\cos t)^{\alpha}\,\mathrm{sign}(\cos t)$$

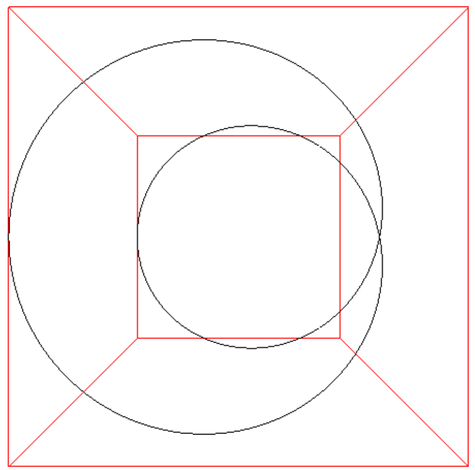
$$y(t) = R\,(\sin t)^{\alpha}\,\mathrm{sign}(\sin t)$$

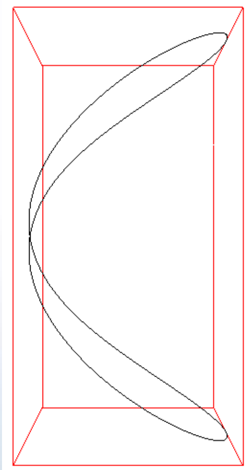
(x0, y0) center

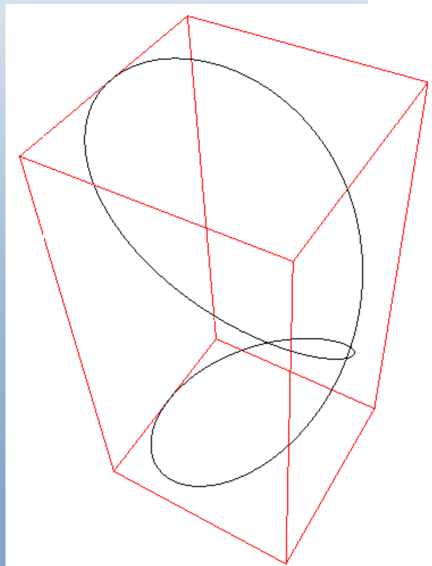
R - radius

$t \in [0, 2\pi]$ angle

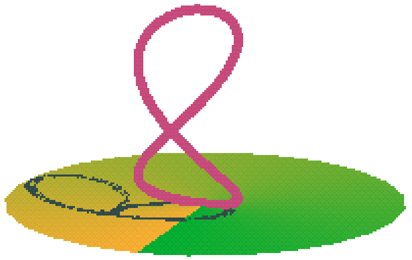
3D Viviani curve







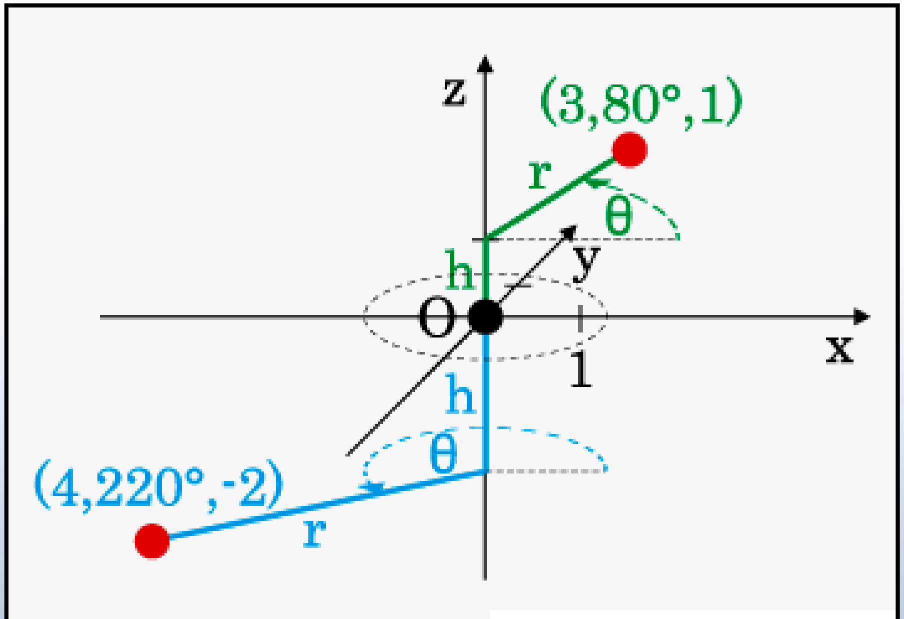
$$x = R\,(1 + \cos(t))$$
$$y = R \sin(t)$$
$$z = 2R \sin(t/2)$$
$$-2\pi < t < 2\pi$$



Images by
Paul Bourke

Animation by Vladimir Rovenski

Cylindrical coordinates



$$x = r\cos\theta \qquad y = r\sin\theta \qquad z = h$$

$$r = \sqrt{x^2 + y^2} \qquad \theta = \arctan\frac{y}{x} \qquad h = z$$

A point P in 3D space is represented by a tuple of three components (*r*, *θ*, *h*):

$r \geq 0$ is the distance from the origin to the point P;

$0 \leq \theta \leq 360^o$ is the angle between the polar axis and the line from the origin to the point P;

h (height) is the signed distance from *xy*-plane to the point P.

Helix

Cylindrical system:

$$r = R$$
$$\theta = t$$
$$h = t$$

Cartesian system:

$$x = R \cos t$$
$$y = R \sin t$$
$$z = t$$

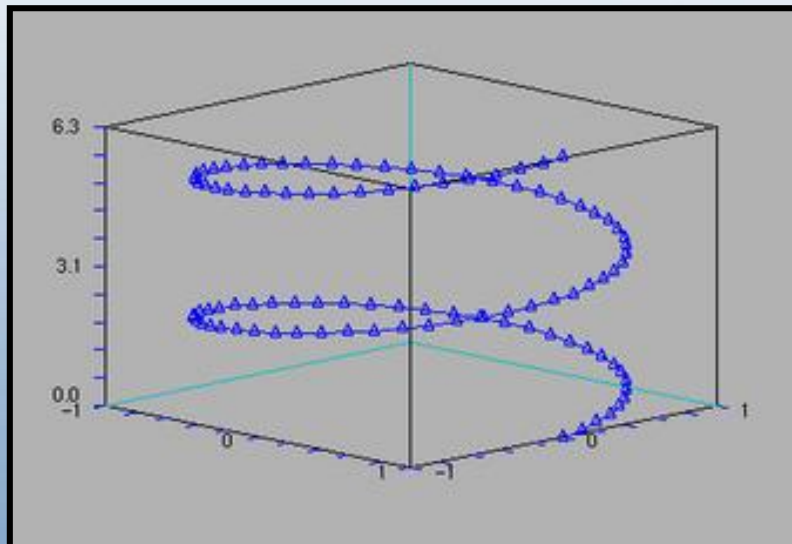
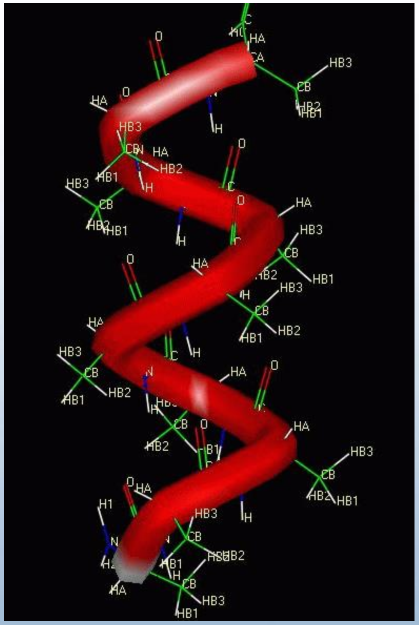


Image by Stéphane Mottelet



Structural Elements of Protein
www.imb-jena.de

Interpolation and Approximation

Curve fitting is a method of constructing new data points from a discrete set of known data points *(P0, P1, …,Pk).*

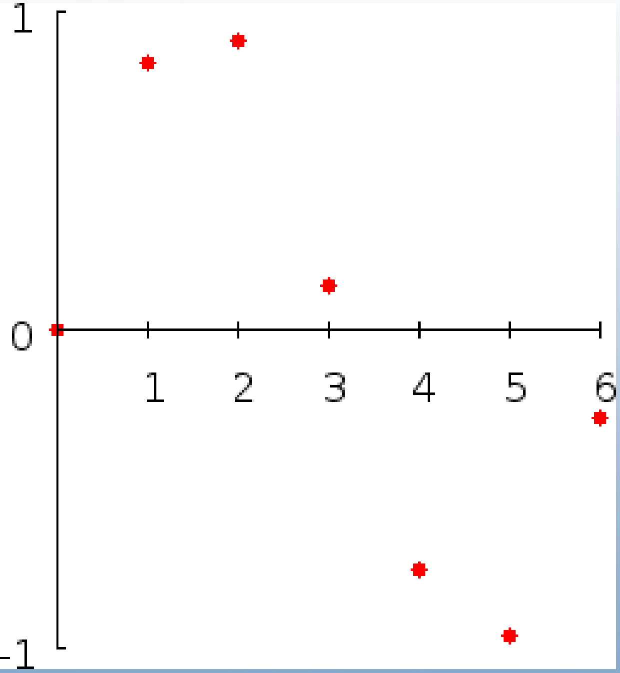
The problem is to find a curve *P(u)* which closely fits the data points.

Interpolation is a specific case of curve fitting, in which the curve must go exactly through the data points.

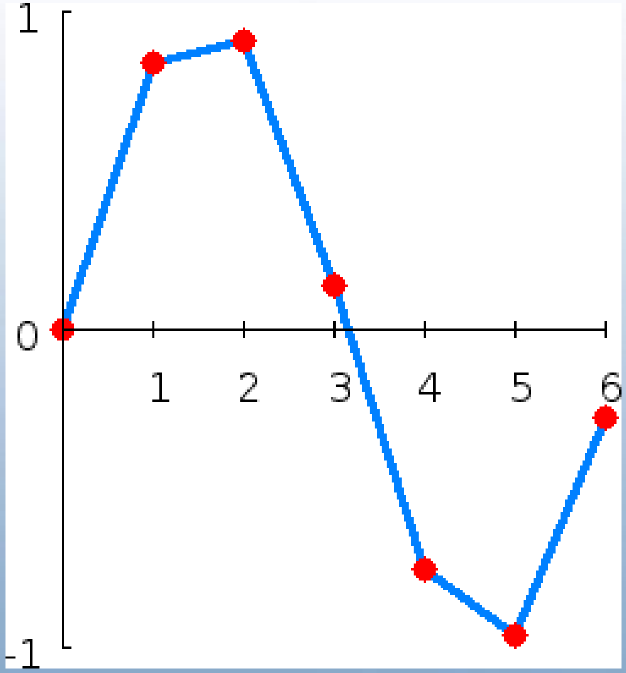
Approximation *curve* passes near the data (control) points, only endpoints are interpolated.

Interpolation problem

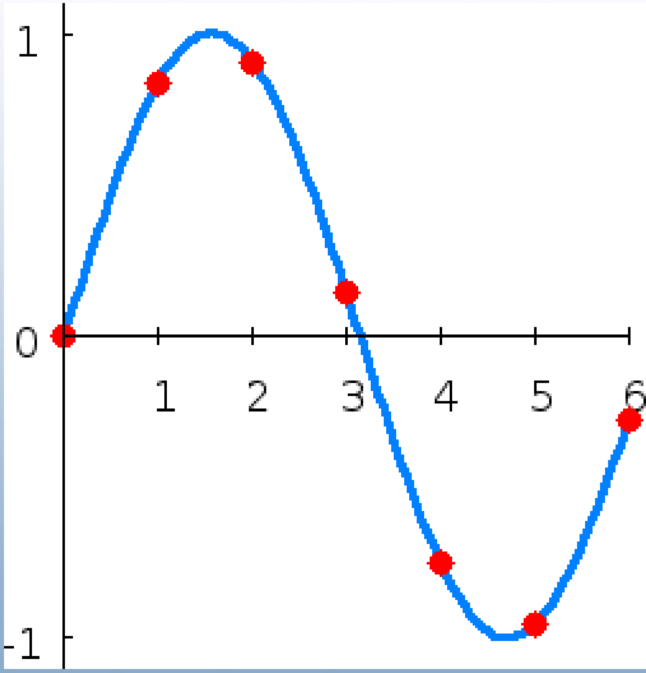
Data points



Linear interpolation



Smooth interpolation

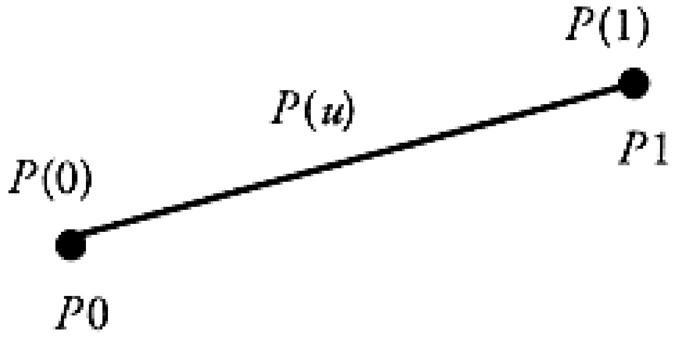


Linear interpolation

Geometric form:

$$P(u) = (1-u) \cdot P0 + u \cdot P1$$

$$P(u) = F_0(u) \cdot P0 + F_1(u) \cdot P1$$



where $F_0(u)$ and $F_1(u)$ are *blending functions*.

Algebraic form:

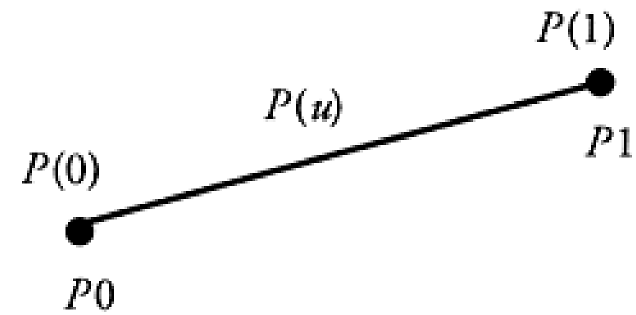
$$P(u) = (P1 - P0) \cdot u + P0$$

$$P(u) = a1 \cdot u + a0$$

Matrix representation

Geometric form:

$$P(u) = \begin{bmatrix} F_0(u) \\ F_1(u) \end{bmatrix} \begin{bmatrix} P0 & P1 \end{bmatrix} = FB^T$$



Algebraic form:

$$P(u) = \begin{bmatrix} u & 1 \end{bmatrix} \begin{bmatrix} a1 \\ a0 \end{bmatrix} = U^T A$$

$$P(u) = \begin{bmatrix} u & 1 \end{bmatrix} \begin{bmatrix} -1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} P0 \\ P1 \end{bmatrix} = U^T MB = FB = U^T A$$

Form for parametric curves of any polynomial order

Interpolating curves

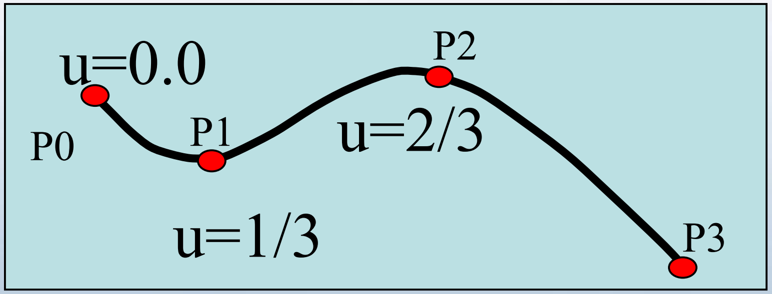
- Four-point form
- Hermite interpolation
- Catmull-Rom spline
- Bézier spline
- B-splines

Splines

A *spline* is a mathematical technique for generating a single geometric object from pieces.

Changes to one piece of the curve do not have significant effects on remote pieces.

To define a spline curve for a range of values for the parameter $u \in [0,1]$, one needs to assign curve pieces to the three intervals [0,1/3], [1/3,2/3], [2/3, 1].

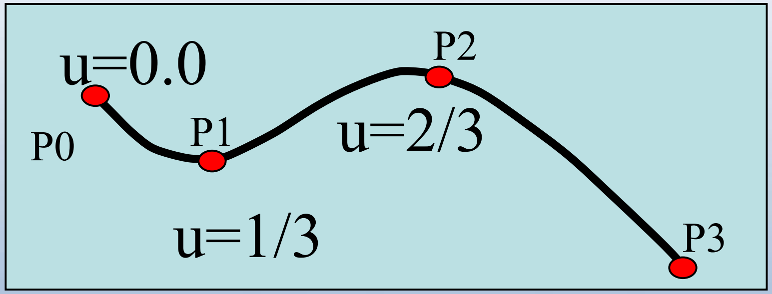


Four-point form

Fitting a cubic segment to four points

$x = f(u)$
$y = g(u)$
$z = h(u)$

Parametric form: $P = P(u) = (x,y,z)$



Space-curve

$P = P(u) \quad 0.0 \leq u \leq 1.0$

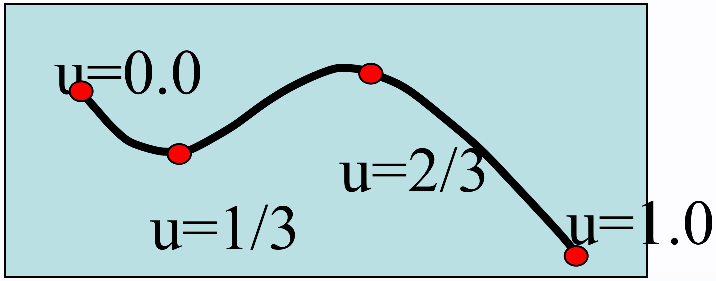
Equations to determine coefficients c_k:

$P(0) = P0$

$P(1/3) = P1$

$P(2/3) = P2$

$P(1) = P3$



Four-point form

$$p(u) = \sum_{k=0}^{3} c_k u^k$$

- Four coefficients to determine for each of x, y and z

$$P(u) = a*u^3 + b*u^2 + c*u + d$$

$$P(0.0) = d = P0$$

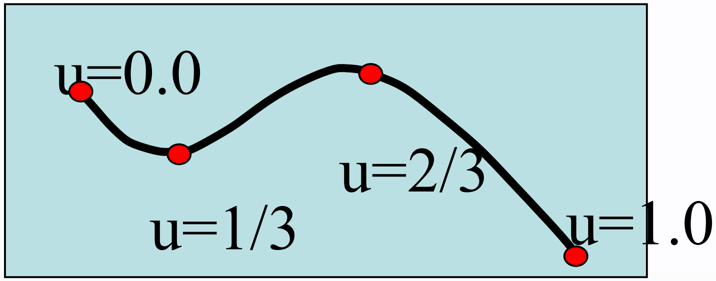
$$P(1/3) = a*(1/3)^3 + b*(1/3)^2 + c*(1/3) + d = P1$$

$$P(2/3) = a*(2/3)^3 + b*(2/3)^2 + c*(2/3) + d = P2$$

$$P(1.0) = a + b + c + d = P3$$

System of linear equations for the coefficients of the cubic polynomials for each of coordinates (x,y,z)

Four-point form



Matrix form for a cubic parametric segment *P(u)* fitting four given points

$$P(u) = U^T M B$$

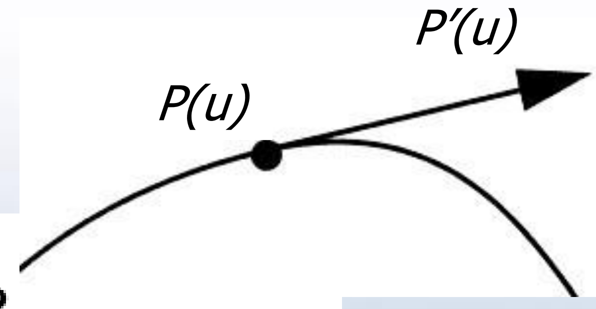
$$U^T = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix}$$

$$M = \frac{1}{2}\begin{bmatrix} -9 & 27 & -27 & 9 \\ 18 & -45 & 36 & -9 \\ -11 & 18 & -9 & 2 \\ 2 & 0 & 0 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} P_{i-1} \\ P_i \\ P_{i+1} \\ P_{i+2} \end{bmatrix}$$

Problem: difficult to join such neighboring segments with C^1 continuity

Derivatives of a cubic curve

Derivatives are necessary to specify tangent vectors for the curves of degree higher than 1.



For a cubic curve:

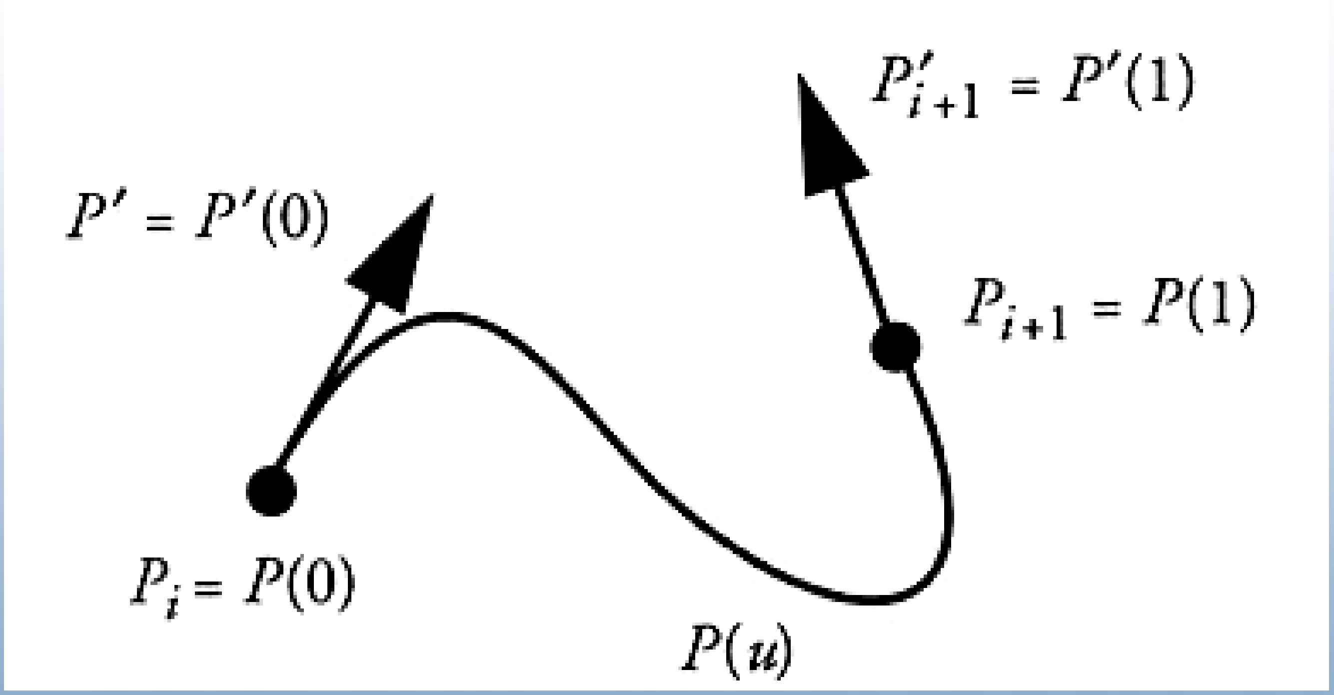
$$P(u) = U^T MB = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} MB$$

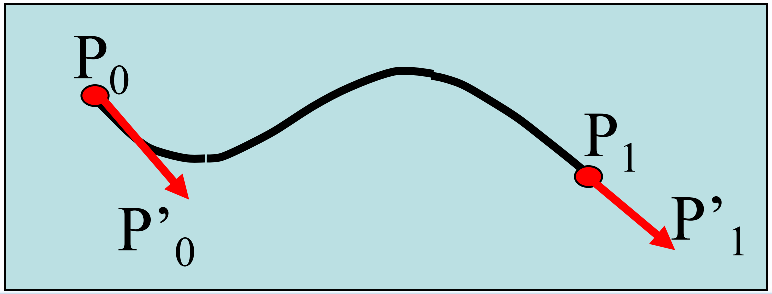
$$P'(u) = U'^T MB = \begin{bmatrix} 3 \cdot u^2 & 2 \cdot u & 1 & 0 \end{bmatrix} MB$$

$$P''(u) = U''^T MB = \begin{bmatrix} 6 \cdot u & 2 & 0 & 0 \end{bmatrix} MB$$

Hermite interpolation

Given data: points + tangent vectors





Hermite interpolation

$$P(u) = a*u^3 + b*u^2 + c*u + d$$

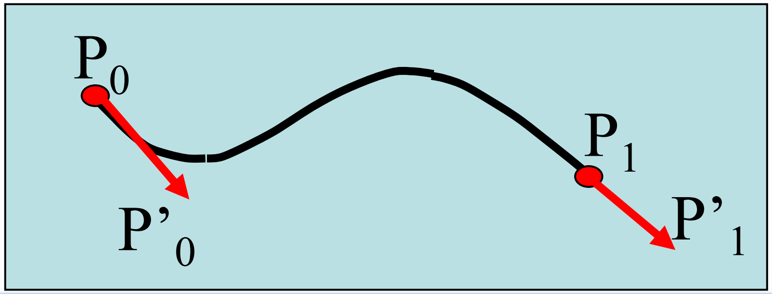
$$P(0.0) = d = P_0$$

$$P(1.0) = a + b + c + d = P_1$$

$$P'(0.0) = c = P'_0$$

$$P'(1.0) = 3*a + 2*b + c = P'_1$$

System of linear equations for the coefficients of the cubic polynomials for each of coordinates (x,y,z)



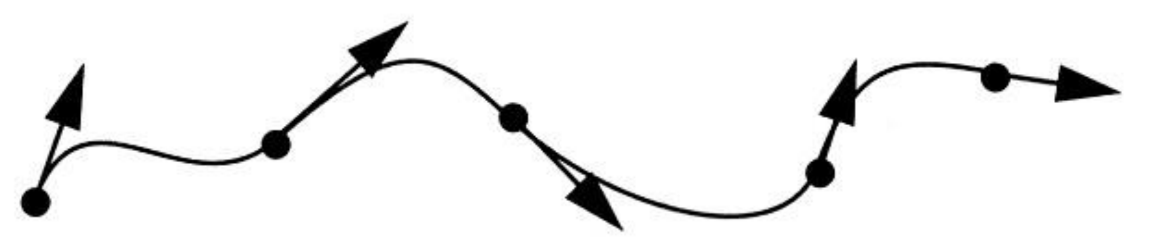
Hermite interpolation

$$P(u) = U^T M B$$

Matrix form for a Hermit segment *P(u)*

$$U^T = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix}$$

$$M = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} P_i \\ P_{i+1} \\ P'_i \\ P'_{i+1} \end{bmatrix}$$



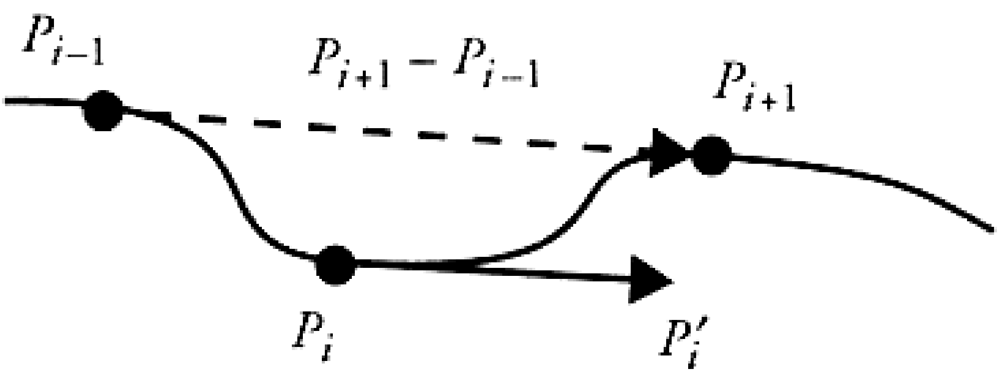
Composite

Hermite curve

Catmull-Rom spline

This spline can be viewed as a Hermite curve, in which the tangent vectors at the internal points are automatically generated

$$P'_i = (1/2) \cdot (P_{i+1} - P_{i-1})$$



Catmull-Rom Spline

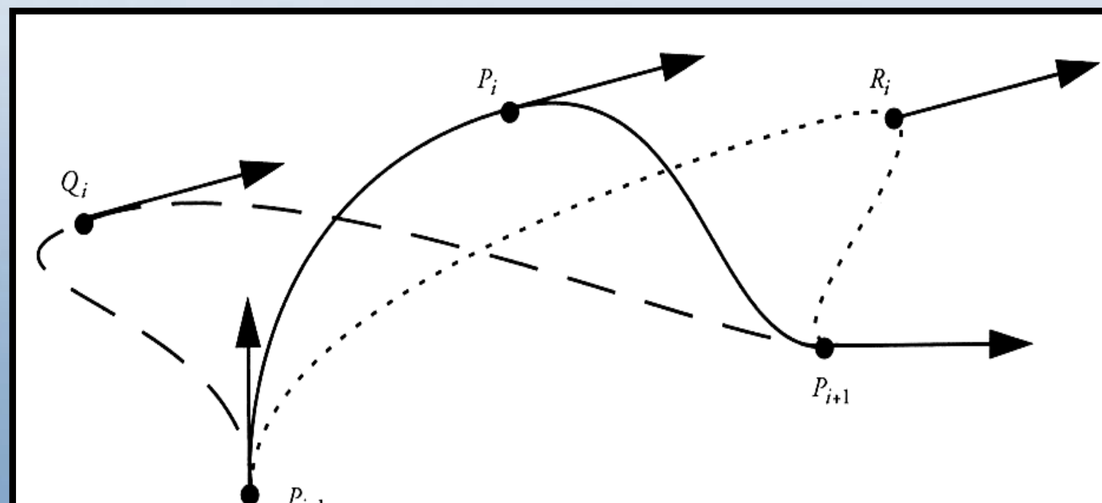
$$P(u) = U^T M B$$

$$U^T = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix}$$

$$M = \frac{1}{2}\begin{bmatrix} -1 & 3 & -3 & 1 \\ 2 & -5 & 4 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 0 \end{bmatrix}$$

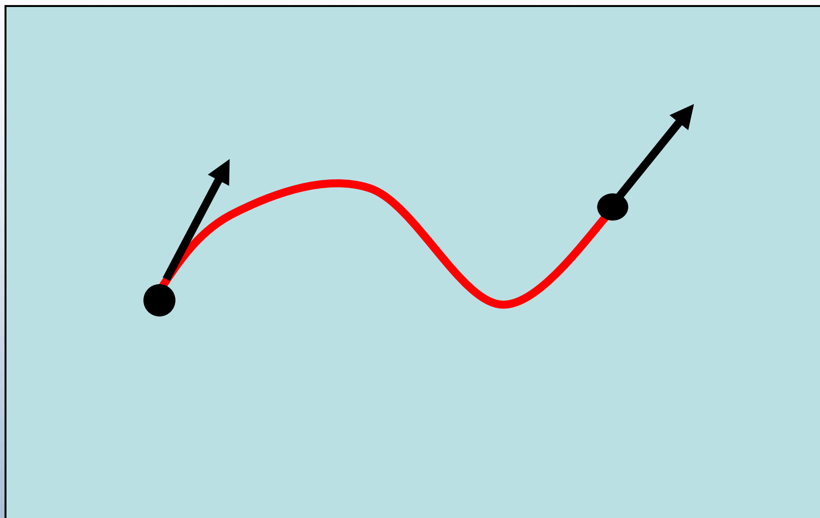
$$B = \begin{bmatrix} P_{i-1} \\ P_i \\ P_{i+1} \\ P_{i+2} \end{bmatrix}$$

The tangent vectors at the end points can be provided by the user or calculated automatically.

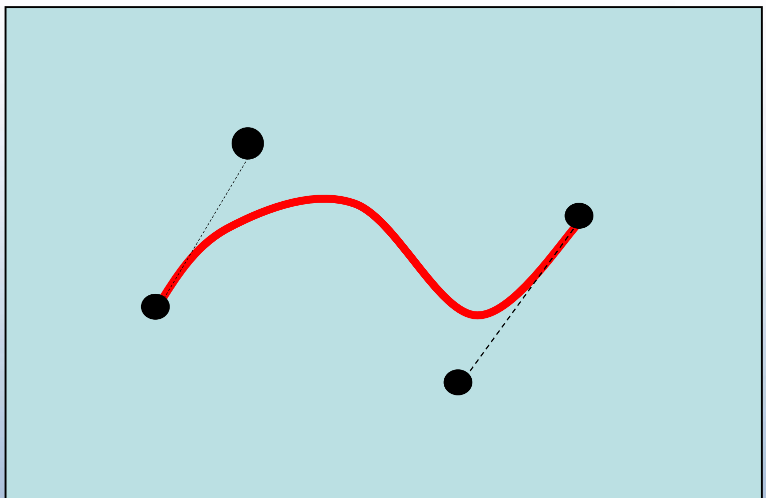


Bézier spline

Hermit segment

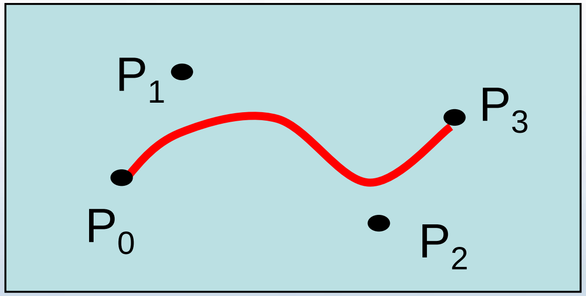


Bézier segment



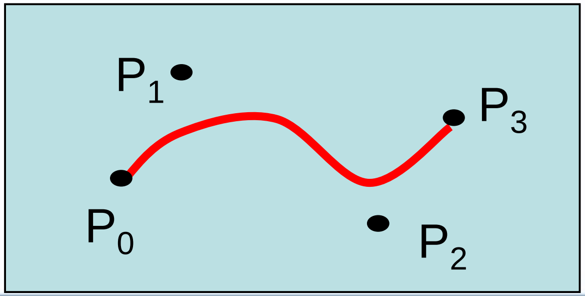
The Bezier form uses two additional points to define tangent vectors at the ending points.

A cubic Bézier curve is defined by the beginning and ending points P_0 and P_3 (interpolated) and two interior points P_1 and P_2 (shape control)



The Bézier curve uses auxiliary control points P_1 and P_2 to define tangent vectors at P_0 and P_3 respectively

$$P'(0) = P_1 - P_0$$
$$P'(1) = P_3 - P_2$$



Bézier spline

$$P(u) = U^T M B$$

$$U^T = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix}$$

Matrix form for a cubic Bézier curve

$$B = \begin{bmatrix} P_{i-1} \\ P_i \\ P_{i+1} \\ P_{i+2} \end{bmatrix}$$

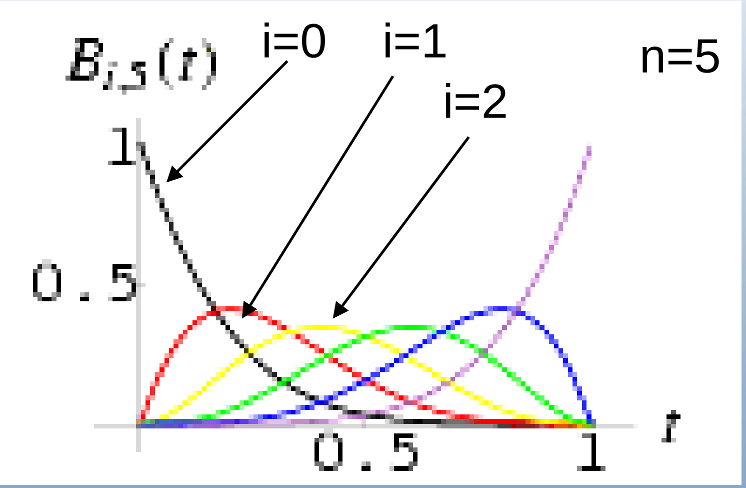
$$M = \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

Bézier spline for n control points P_i

$$\mathbf{C}(t) = \sum_{i=0}^{n} \mathbf{P_i} B_{i,n}(t),$$

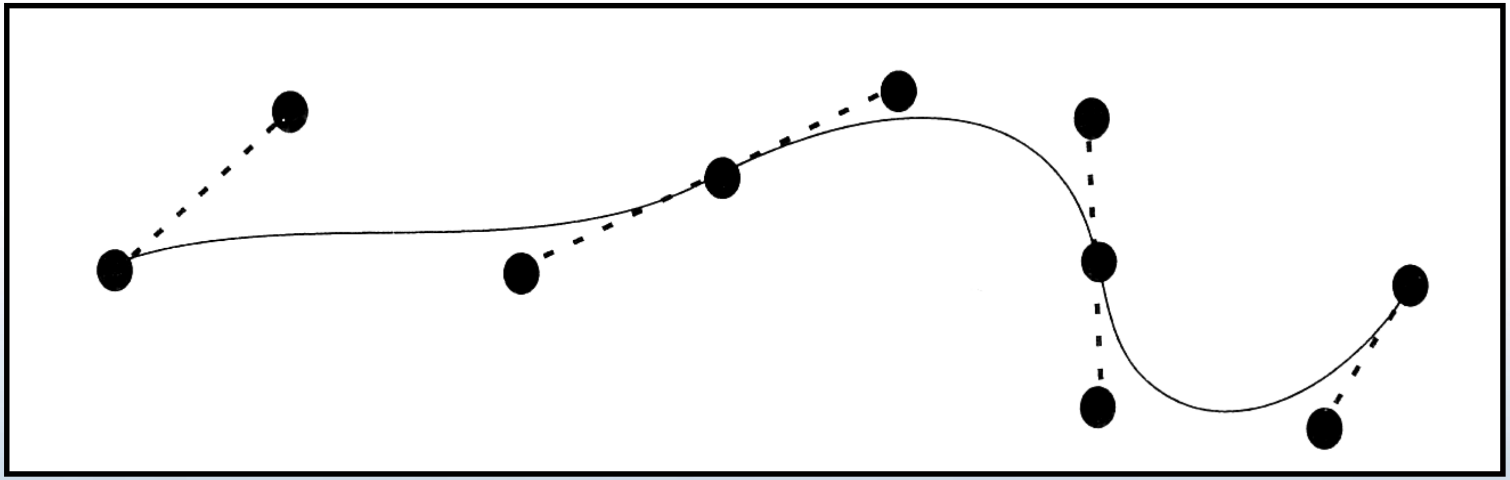
Where $B_{i,n}(t)$ are weighting functions called Bernstein polynomials:

$$B_{i,n}(t) = \binom{n}{i} t^i (1-t)^{n-i}$$



Degree of the polynomial grows with the number of control points.

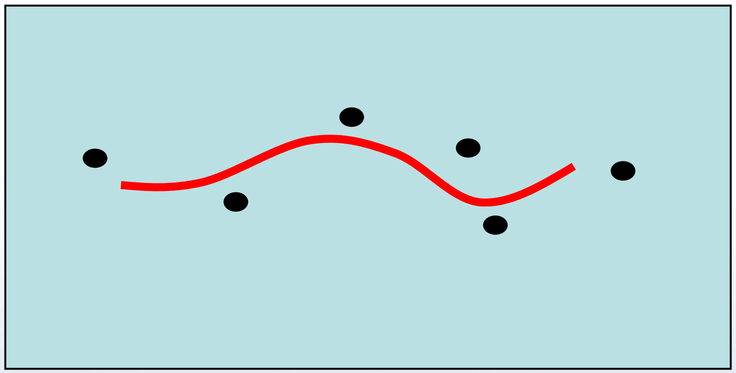
Bézier spline



The Bézier curve always passes through the first and last control points and lies within the <u>convex hull</u> of the control points.

Continuity between adjacent segments in a composite Bézier curve can be controlled by the collinearity of the control points on both sides of a shared endpoint of two segments.

B-spline



B-spline is a generalization of the Bézier spline:

$$\mathbf{C}(t) = \sum_{i=0}^{n} \mathbf{P_i} N_{i,p}(t)$$

where *Pi* are control points and *Ni* are called blending functions.

• Any number of points can be added without increasing the degree of the polynomial.

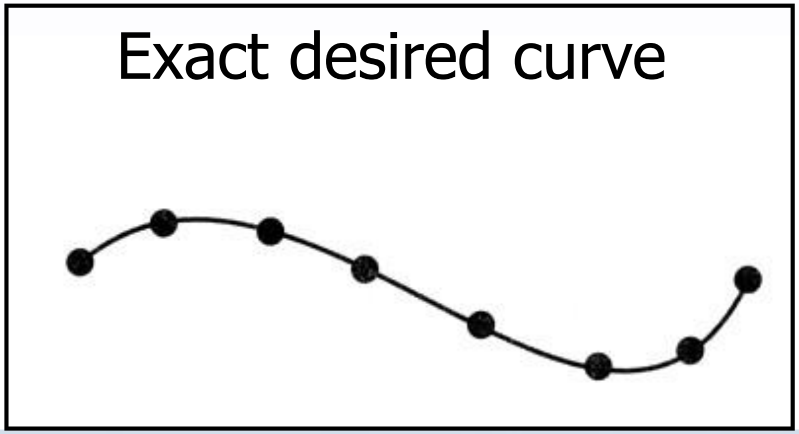
• The spline is completely local - changes to a control point only affects the curve in that locality

• Closed curves can be created by making the first and last points the same, although continuity will not be maintained automatically.

• B-splines lie in the convex hull of the control points.

Requirements to Curves

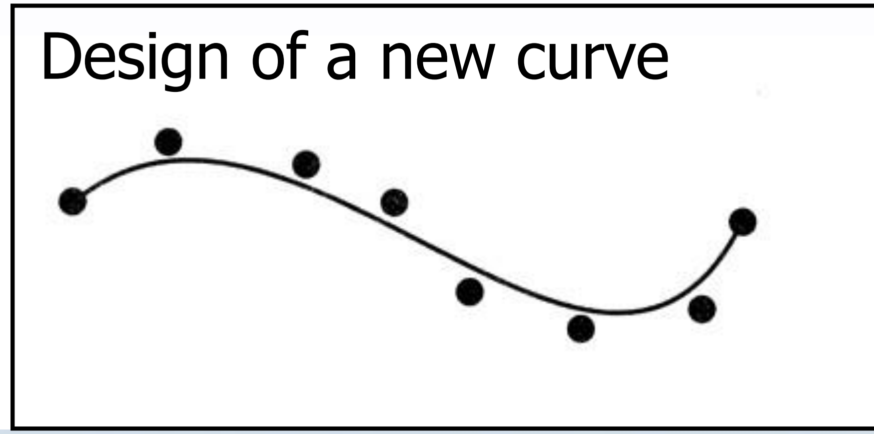
- Interpolation vs Approximation



Interpolating curve passes through the given control points:

Hermite curve,

Catmull-Rom spline

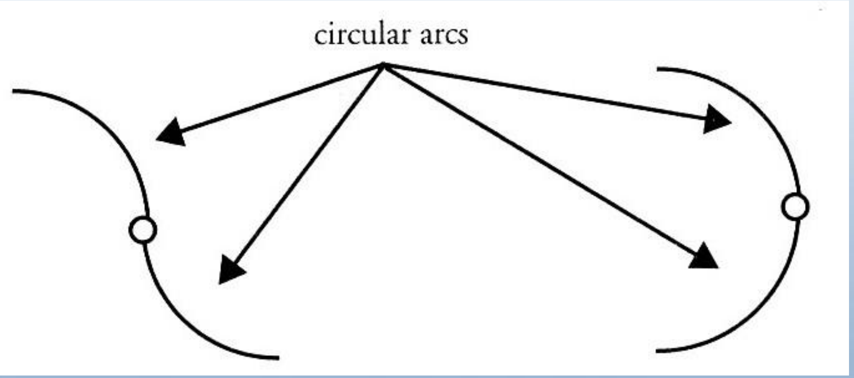


Approximating curve passes near the control points, only endpoints are interpolated:

Bezier spline, B-spline

- Continuity – smoothness of the curve

Positional C^0 discontinuity Tangential C^1 discontinuity



Positional and tangential continuity, curvature discontinuity

Positional, tangential, and curvature continuity

Requirements to Interpolation

- Continuity

C^1 continuity

Hermite curve, Catmull-Rom spline
parabolic blending, cubic Bezier curve

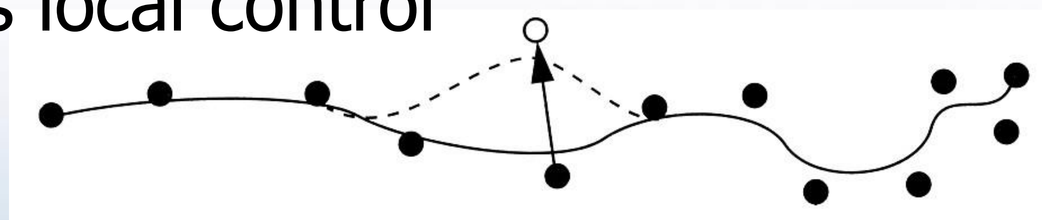
C^2 continuity

compound Hermite curve, B-spline

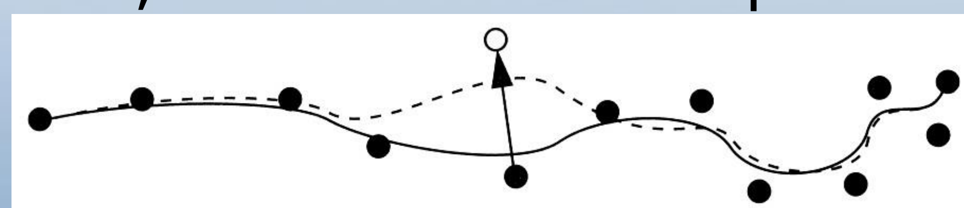
- Complexity – influences computation time.

Cubic polynomials are the lowest order polynomials.

- Global vs local control



Local control: moving one point changes the curve locally: Catmull-Rom splines, cubic Bezier and B-splines – more desirable



Global control: moving one point changes the entire curve: Hermite curve with second-order continuity, higher order Bezier and B-splines

Contents

- *Parametric curves*
- *Polar coordinates*
- *Cylindrical coordinates*
- *Interpolation and approximation*
- ***Parametric surfaces***
- *Spherical coordinates*
- *Trimmed parametric surfaces*

Parametric surface notion

A parametric surface is defined by a mapping of a unit square to n-D space

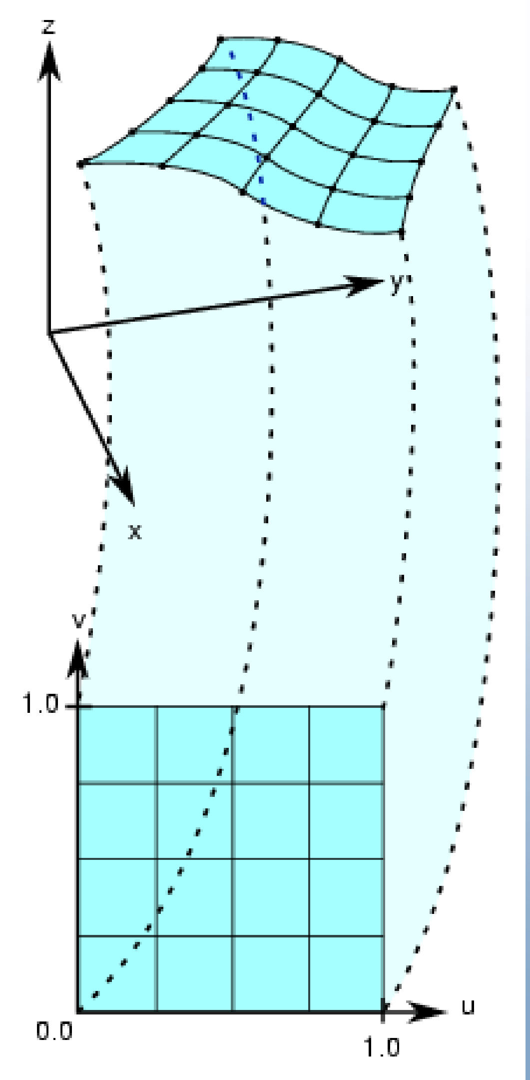
Parametric equations of a surface are obtained by introducing two more extra variables (*u,v*), or parameters, and calculating n-D point coordinates as functions of the parameters *u* and *v* :

$$x_1 = \varphi_1(u,v)$$
$$x_2 = \varphi_2(u,v)$$
$$\ldots =$$
$$x_n = \varphi_n(u,v)$$

Surface in 3D space

Each component of a point on the surface is a function of *u* and *v* which both lie in the parameter interval [0, 1] on the real line. The point (*u,v*) lies in the **unit square** on the *uv*-plane. Points on the surface are described by three functions:

$$(x(u,v),\ y(u,v),\ z(u,v))$$



Parametric plane

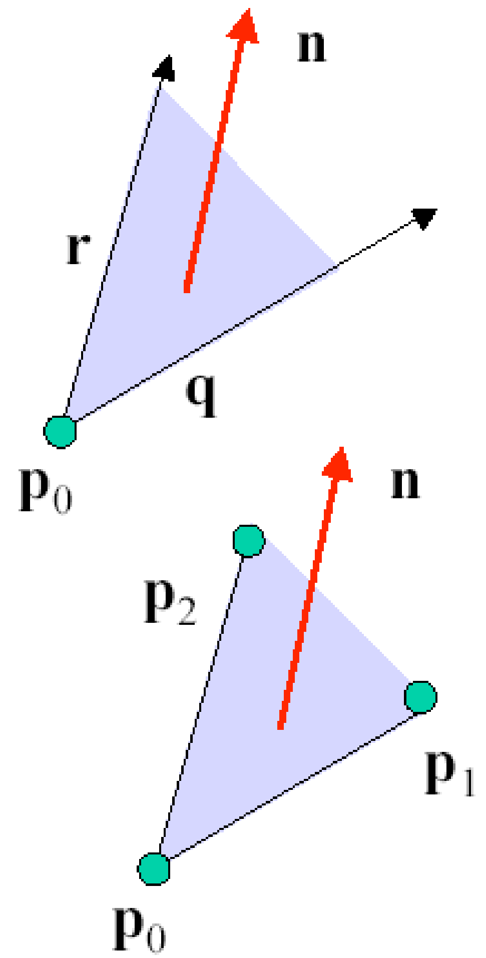
point-vector form

$$\mathbf{p}(u,v)=\mathbf{p}_0+u\mathbf{q}+v\mathbf{r}$$

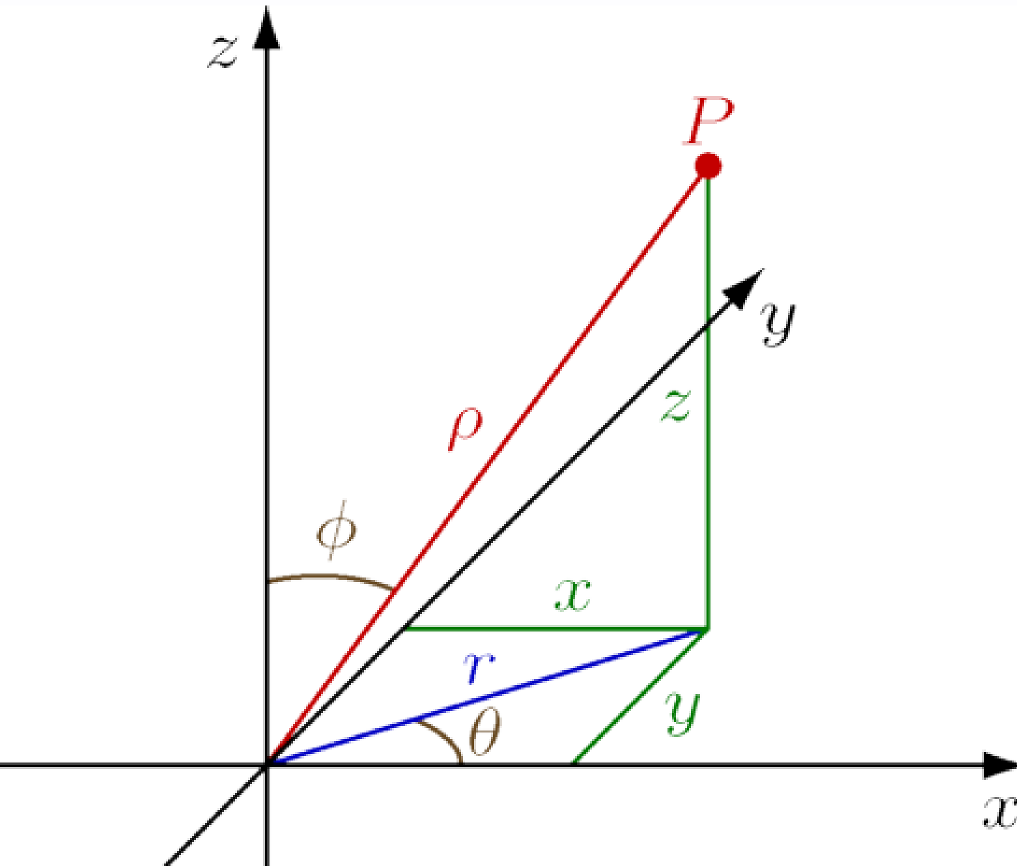
$$\mathbf{n} = \mathbf{q} \times \mathbf{r}$$

three-point form

$$\mathbf{q} = \mathbf{p}_1 - \mathbf{p}_0$$
$$\mathbf{r} = \mathbf{p}_2 - \mathbf{p}_0$$



Spherical coordinates



Point *P* is represented by a tuple of three components (ρ,φ,θ).

$$0 \leq \rho$$

radius is the distance between the point *P* and the origin,

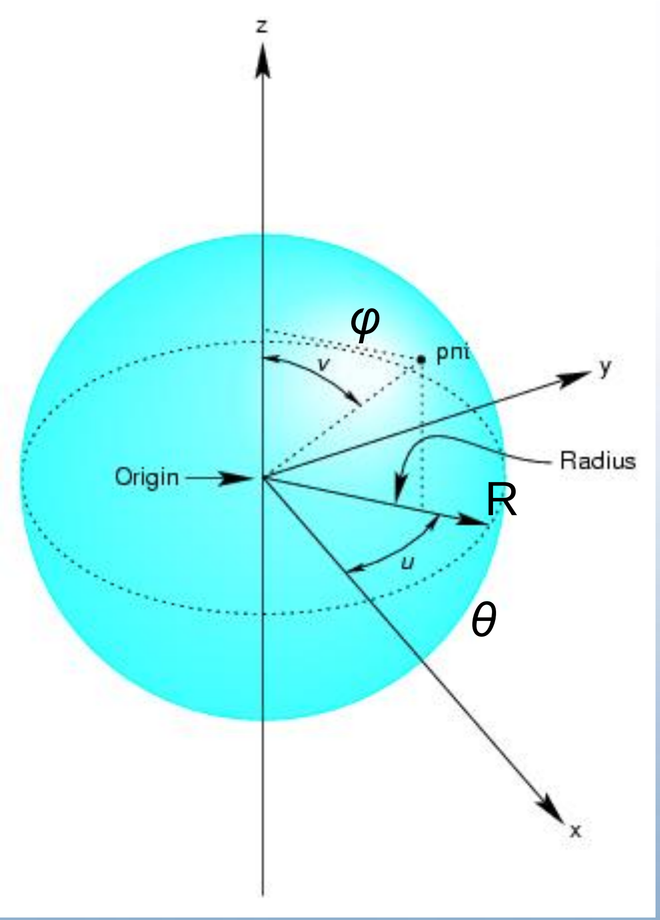
$$0 \leq \phi \leq 180^\circ$$

is the angle between the *z*-axis and the line from the origin to the point P,

$$0 \leq \theta < 360^\circ$$

is the angle between the positive *x*-axis and the line from the origin to the point P projected onto the *xy*-plane.

Parametric Sphere Model



In spherical coordinates:

$$\rho = R$$

Parametric form:

$$x(u,v) = r \cos\theta \sin\phi$$
$$y(u,v) = r \sin\theta \sin\phi$$
$$z(u,v) = r \cos\phi$$

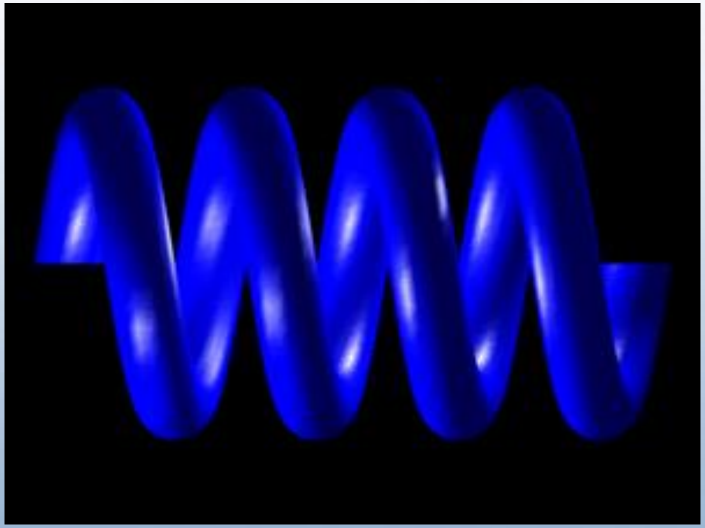
$$360 \geq \theta \geq 0$$
$$180 \geq \phi \geq 0$$

θ constant: circles of constant longitude

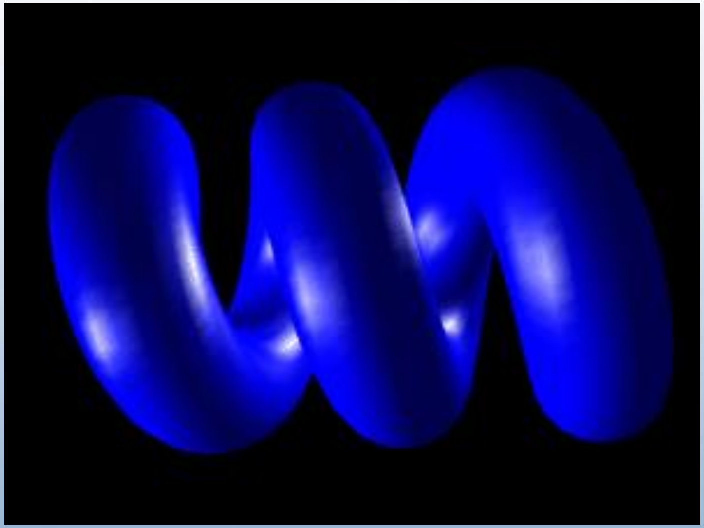
ϕ constant: circles of constant latitude

Spring

x = [1 - r1 * cos(v)] * cos(u)
y = [1 - r1 * cos(v)] * sin(u)
z = r2 * [sin(v) + periodlength * u / pi]



r1 = 0.25, r2 = 0.25,
periodlength=3.0



r1 = 0.5, r2 = 0.5,
periodlength=1.5

Cubic Polynomial Surfaces

$$\mathbf{p}(u,v)=[x(u,v), y(u,v), z(u,v)]^T$$

where

$$p(u,v) = \sum_{i=0}^{3}\sum_{j=0}^{3} c_{ij} u^i v^j$$

p is any of x, y or z

Need 48 coefficients (3 independent sets of 16) to determine a surface patch

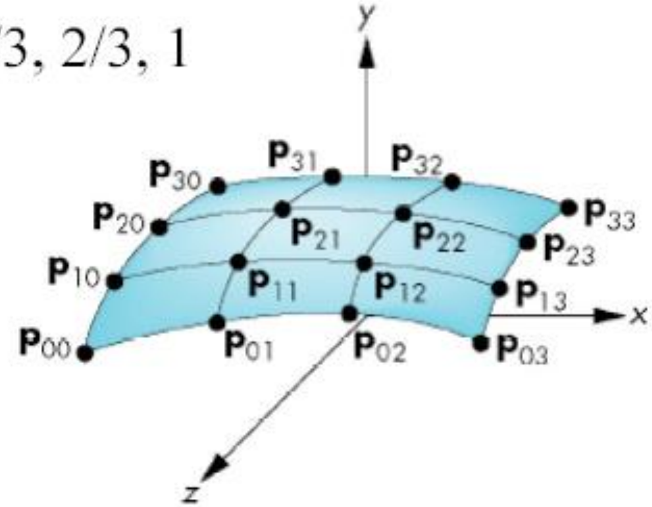
- Interpolating surface patch
- Bezier patch
- B-spline patch

Interpolating Surface Patch

$$p(u,v) = \sum_{i=o}^{3} \sum_{j=0}^{3} c_{ij} u^i v^j$$

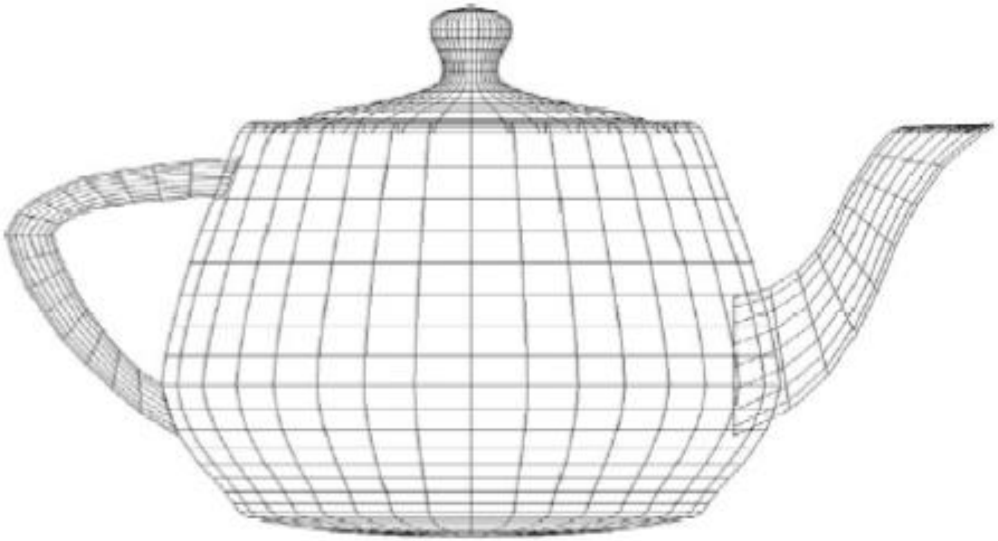
Need 16 conditions to determine the 16 coefficients c_{ij}

Choose at $u,v = 0,\ 1/3,\ 2/3,\ 1$



Utah Teapot

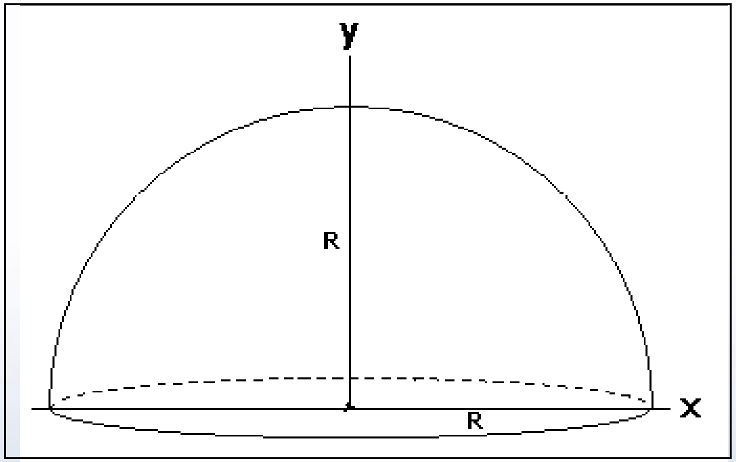
- Most famous data set in computer graphics
- Widely available as a list of 306 3D vertices and the indices that define 32 Bezier patches



Surface with boundary

A surface may have a **boundary**, where the surface ends.

For example, the boundary of a hemisphere would be the circle around the edge.



Trimmed parametric surfaces

A parametric surface with boundary can be trimmed by

- Edges for the surface other than those defined by the *uv* unit square.
- Holes in a surface.
- Defining boundary edges using trim curves and loops.

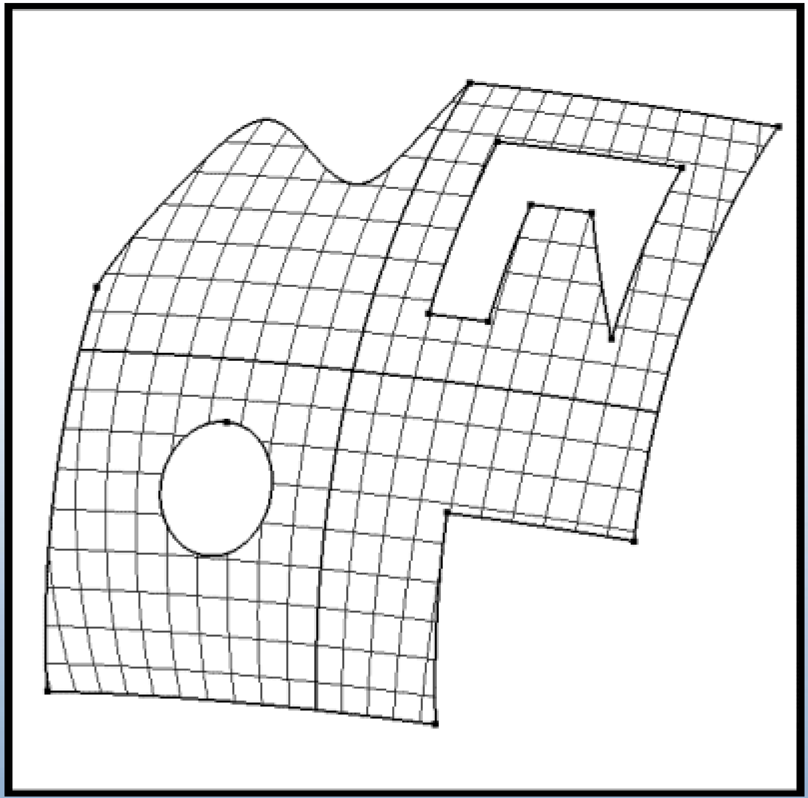
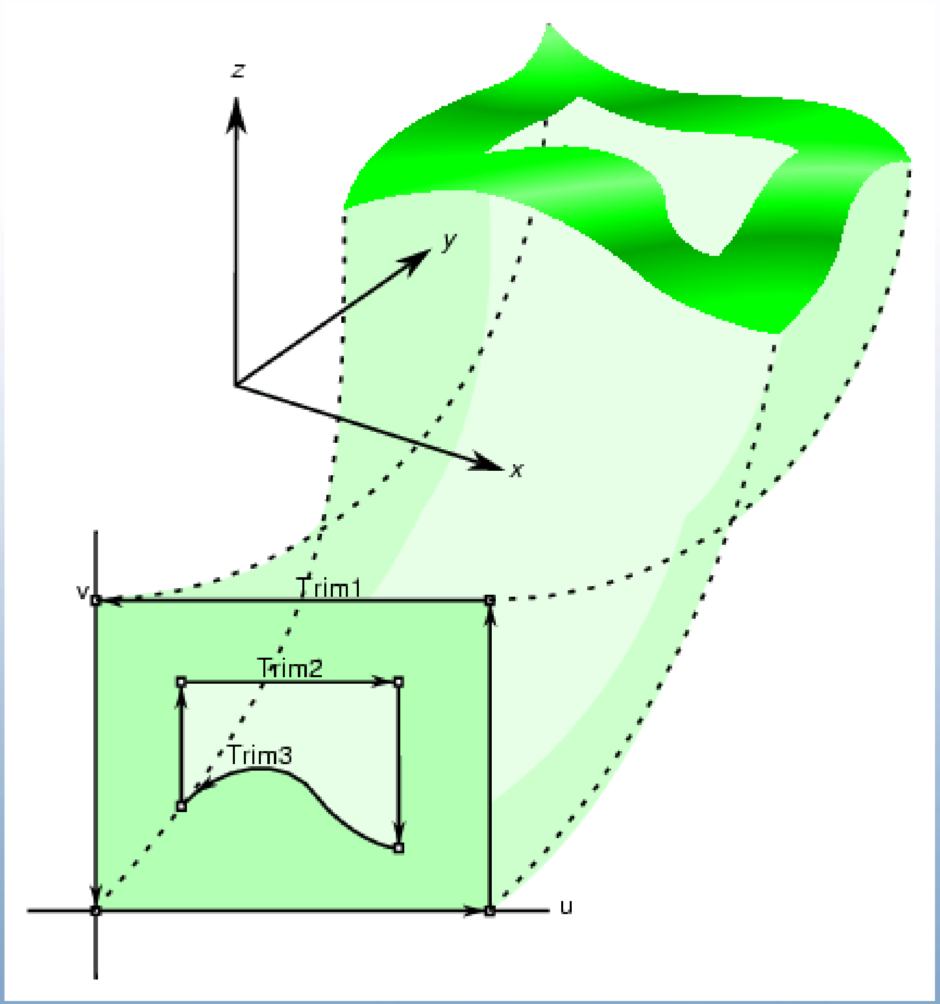


Image by Ken Takusagawa

Trimmed parametric surfaces

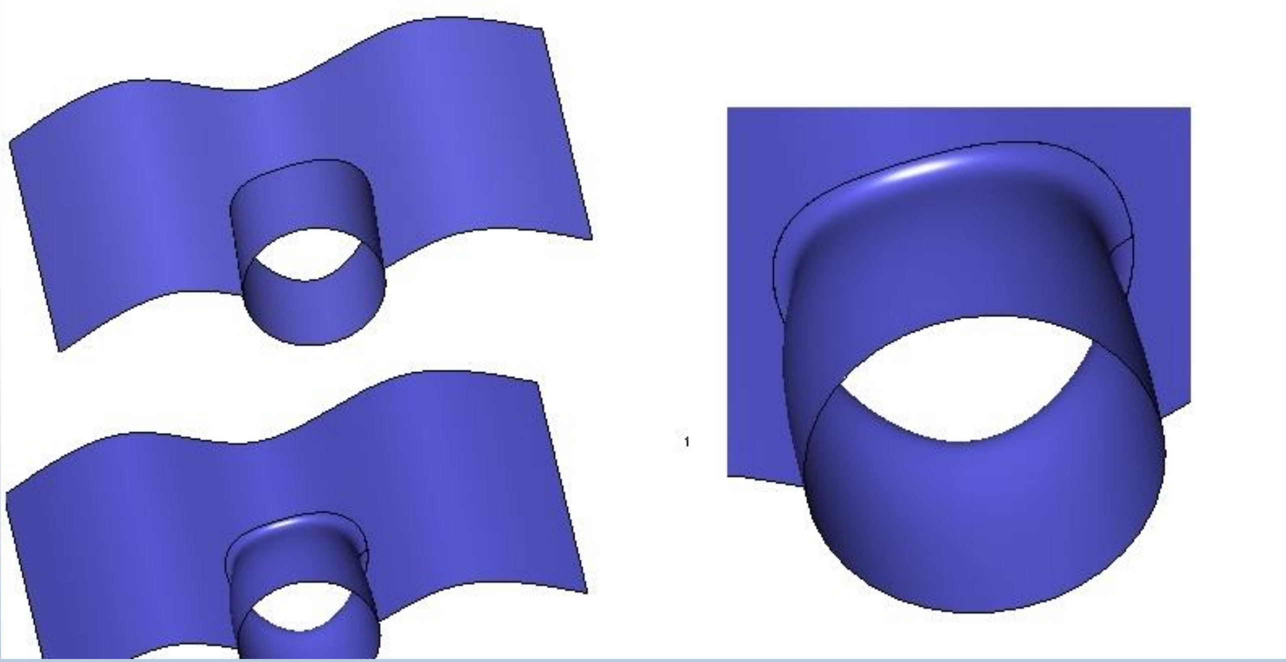
- Trim loops are defined in the *uv*-square and mapped to 3D space
- Left hand rule
- Clockwise loop removes a hole
- Counterclockwise loop keeps the enclosed region and eliminates everything outside.



Trimmed parametric surfaces

Combining two
trimmed surfaces

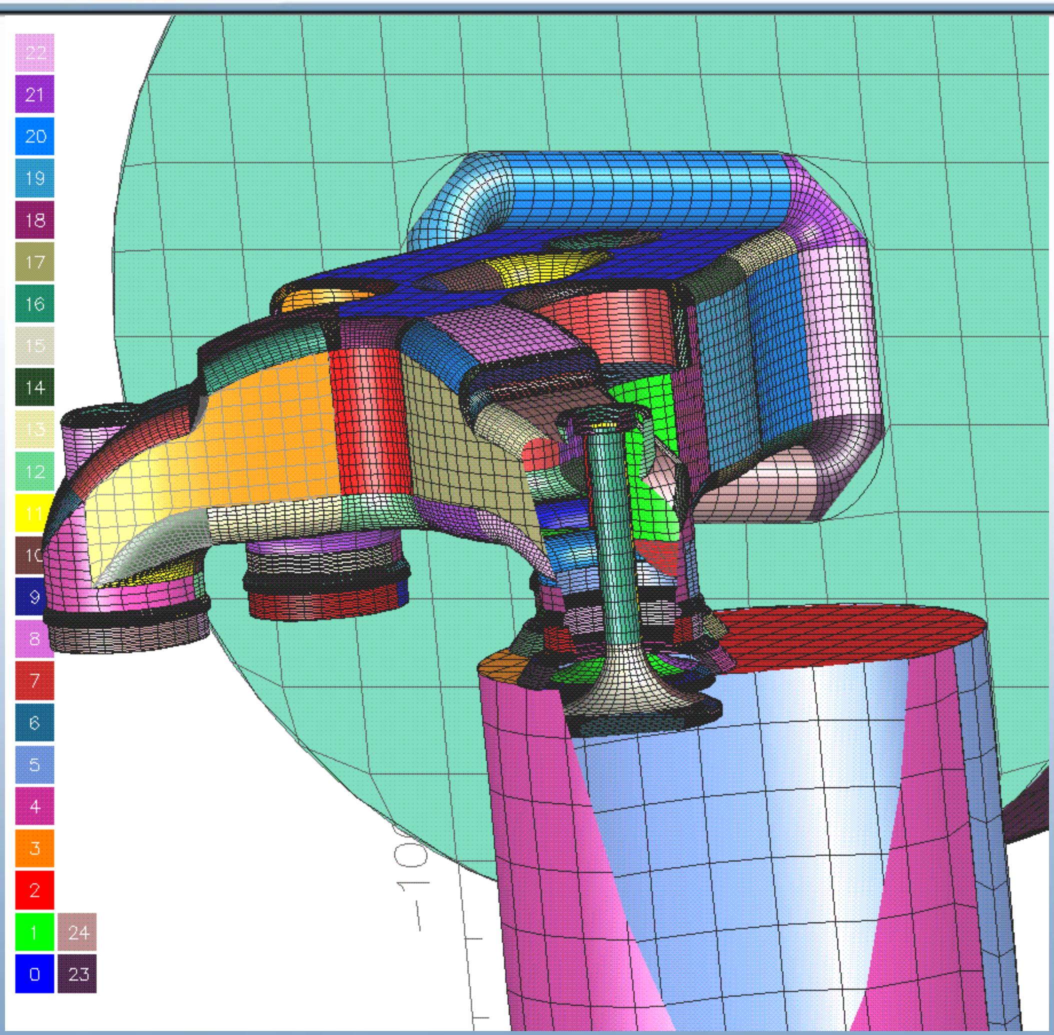
Combining with
continuity matching



www.csi-concepts.com/extreme.htm

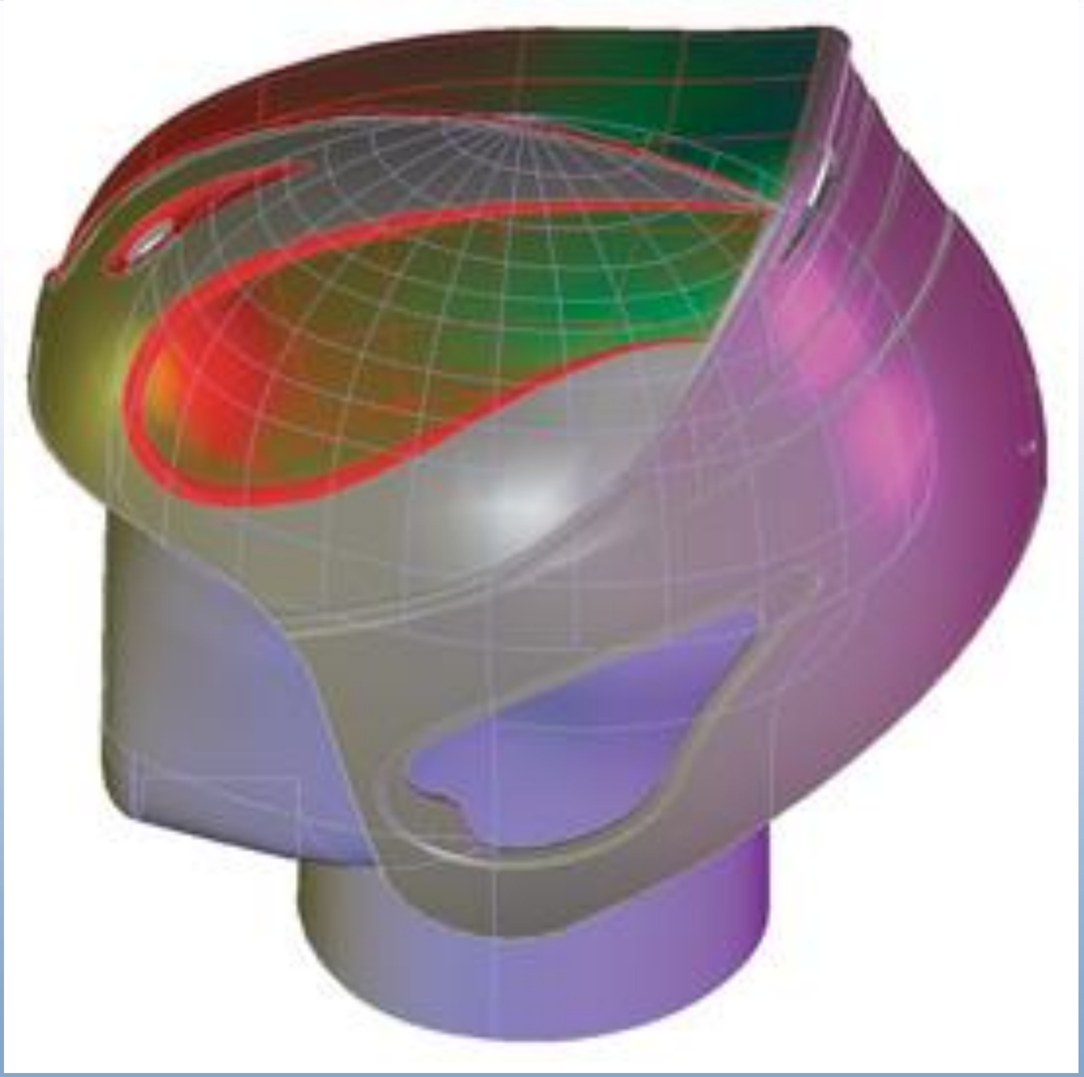
The basic boundary constructing operation for solid modeling

Trimmed parametric surfaces



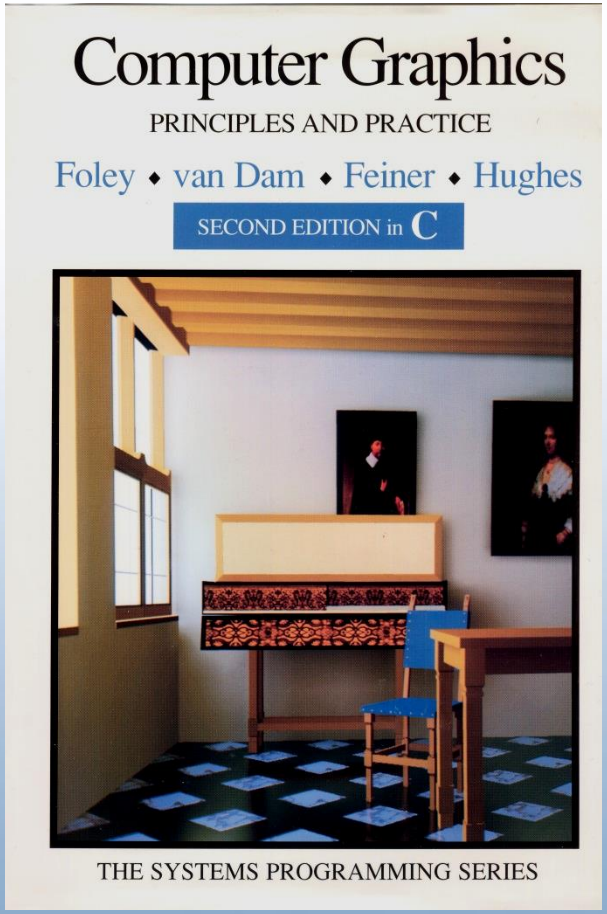
Composite surface of trimmed NURBS surfaces, from proEnginner

The surface model of a racing ski helmet generated in Cadkey Workshop by Louis Garneau Sports Inc., Quebec, Canada, www.louisgarneau.com



References

- James D. Foley, Andries van Dam, Steven K. Feiner, John F. Hughes, Computer Graphics: Principles and Practice (2nd Edition in C), Addison-Wesley, Reading, MA, 1997.



References

- A. Bowyer, J. Woodwark, Introduction to Computing with Geometry, Information Gepmeters, Wincheter, UK, 1993.
- A. Bowyer, Geometric Modelling Course, University of Bath, UK, 1996.
- A. A. G. Requicha, Geometric Modeling: a First Course http://www-pal.usc.edu/~requicha/book.html
- E. Angel, Interactive Computer Graphics, 4th edition, Addison-Wesley, 2005.
- R. Parent, Computer Animation – Algorithms and Techniques, Morgan Kaufmann Publishers, 2002.
- K. Takusagawa, Representing Smooth Surfaces, Lecture, MIT, 2001.
- Wikipedia, www.wikipedia.org
- OpenGL Performer Programmer's Guide, SGI, 2004.