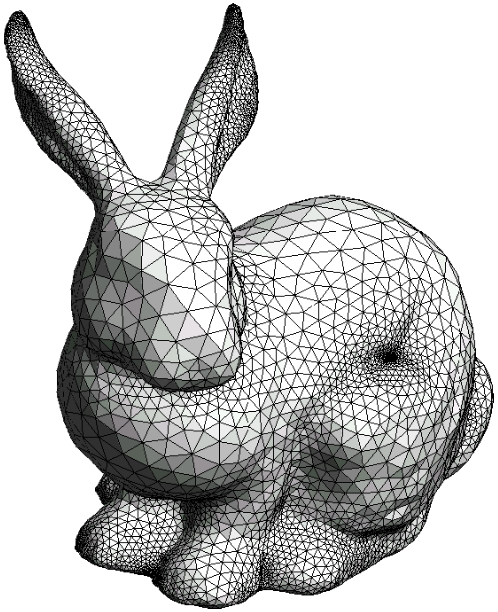


Geometric Modeling in Graphics

Part 6: Curves

Martin Samuelčík
www.sccg.sk/~samuelcik
samuelcik@sccg.sk

Curve

- 1D set of points, embedded in space **X** (E^2, E^3)
- f: **R** → **X**
- Parametric curves
  - Set of all points X ϵ **X** such that X = f(t), t ϵ <a,b>
  - Line: X = S +tD, t ϵ **R**, S - start point, D - direction vector
  - Circle in 2D: X= (r.cos t, r.sin t), t ϵ <0,2π>, r – radius
- Implicit curves
  - Set of all points X ϵ $\mathbf{E}^2$ such that f(X)=0
  - Line: (X-P).N=0, P - any point on line, N - normal of line, inner product
  - Line in 2D: ax+by+c = 0
  - Circle: |X-C|-r=0, C-center, r-radius
  - Circle in 2D: $(x-cx)^2+(y-cy)^2-r=0$

Parametric curve

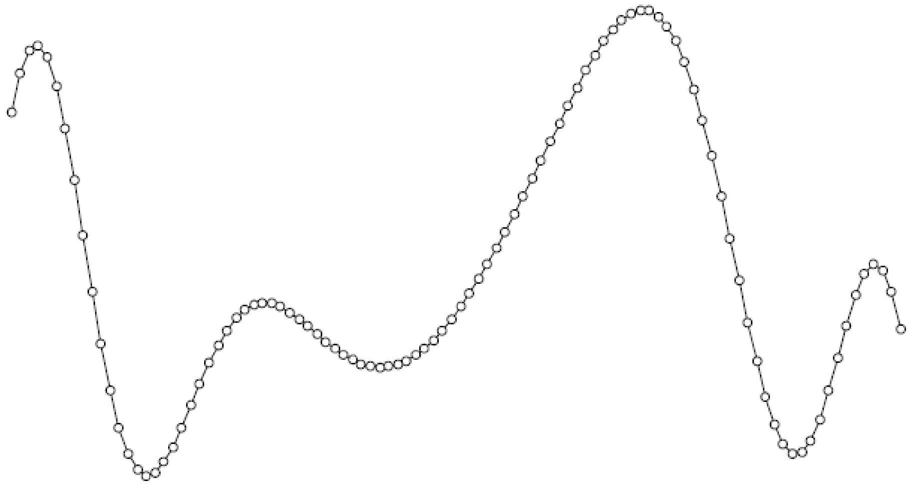
- Suitable for many modeling algorithms
- Given parametrization – easy „walk" on curve, easy to generate points on curve
- Visualization
  - Approximation with piecewise linear curve – polyline
  - Given domain interval <a,b>, choose sample values $a=t_0 < t_1 < t_2 < \ldots < t_m=b$
  - Compute sample curve points $F_0=f(t_0), F_1=f(t_1),\ldots,F_m=f(t_m)$, draw polyline $F_0, F_1,\ldots,F_m$
  - Parameter m – quality of sampling, approzimation, visualization
  - Uniform sampling: $t_i = a+i(b-a)/m, i=0,1,\ldots,m$
  - Adaptive sampling: compute t_i based on curve parameters, for example curvature

Curve adaptive sampling

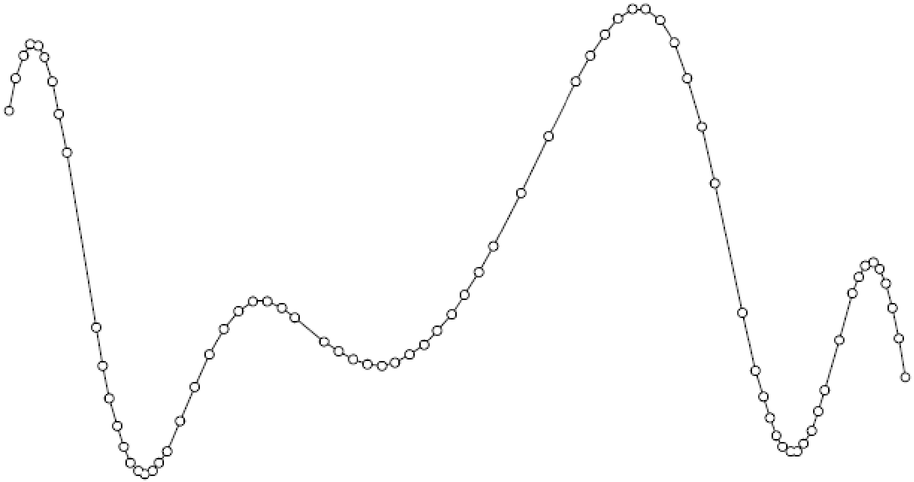
- 1. Starting with domain – interval <a,b>
- 2.For current interval <u,v>, choose value w at random, w=u+d.(v-u), d is picked at random from <0.45,0.55>
  - Store u,v as sampling values
  - Check if curve for <u,v> is flat enough by computing P=f(u), Q=f(v), R=f(w) and using criterion
    - Area of triangle PQR is small
    - Angle PRQ is large enough
    - R is close to chord PQ
    - Tangents of curve at P,Q,R are approximately parallel
  - If curve is not flat enough at <u,v>, divide it into two intervals <u,w>,<w,v> and recursivly call 2. for both
- 3. Organize generated sampling values in one sequence

Parametric curve sampling

https://www.researchgate.net/publication/2757679_IV4_Adaptive_Sampling_of_Parametric_Curves



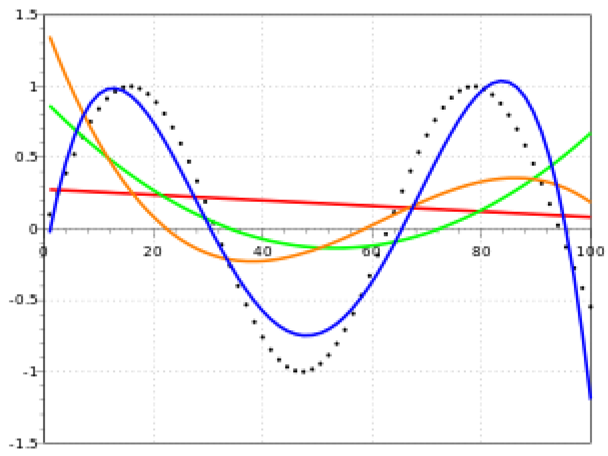
Uniform sampling

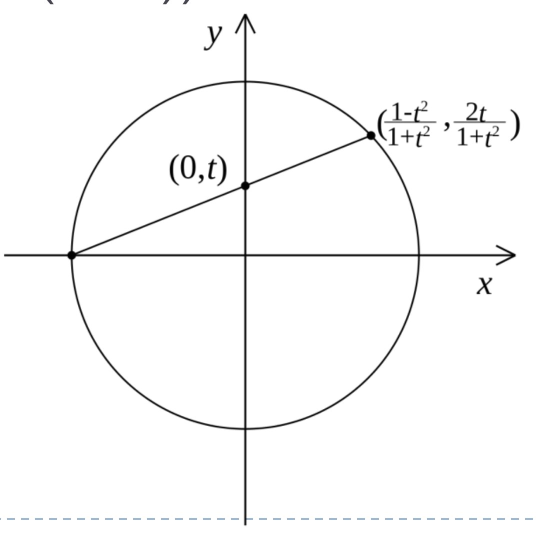


Adaptive sampling

Polynomial curve

- Parametric curve where f is polynomial function
- Popular parametric representation due to fast and easy computation
- In modelling, usually only order up to 3 is used
- Extended to rational curve – fraction of two polynomials
  - Circle in 2D: $f(t)=((1-t^2)/(1+t^2), 2t/(1+t^2)), t \in R$





Polynomial curve

- Several forms of polynomial basis
- Monomial basis
  - $f(t)=V_0+V_1t+V_2t^2+\ldots+V_nt^n,\ t \in \langle a,b\rangle$
  - V_0 - control point, $V_1,\ldots,V_n$ - control vectors
  - Not very suitable for geometric modeling
- Newton, Lagrange interpolation basis
- Bernstein basis, Bezier curve
  - $f(t)=B^n(t)=V_0B^n_0(t)+\ldots V_nB^n_n(t),\ t \in \langle 0,1\rangle$
  - $V_0,V_1,\ldots,V_n$ – control points

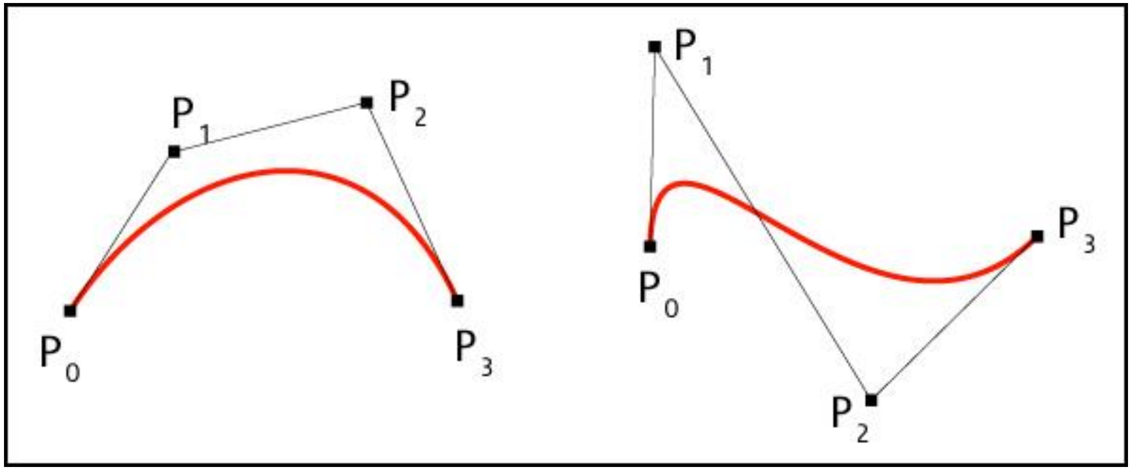
$$B_i^n(t)=\binom{n}{i}t^i(1-t)^{n-i}$$

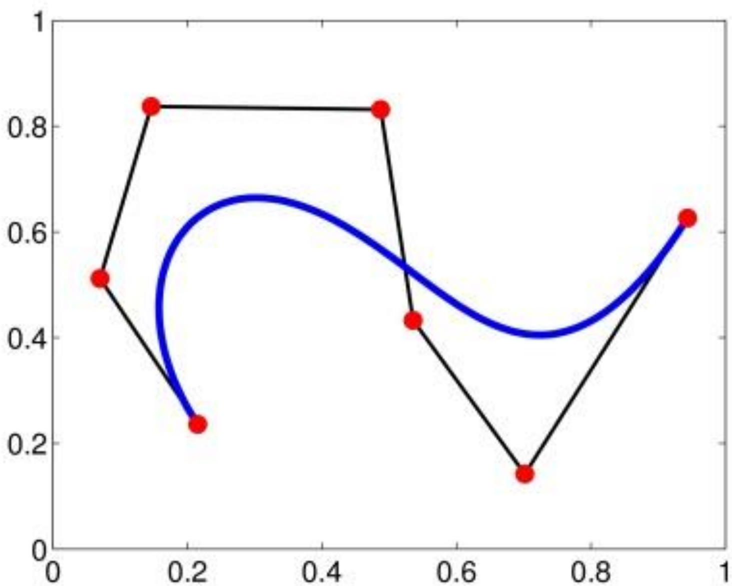
- Hermite basis, Cubic Hermite curve
  - $f(t)=H^3(t)=V_0H^3_0(t)+T_0H^3_1(t)+T_1H^3_2(t)+V_1H^3_3(t),\ t \in \langle 0,1\rangle$
  - V_0,V_1 - interpolated control points, T_0,T_1 – tangent vectors
  - $H^3_0(t)=2t^3-3t^2+1,\ H^3_1(t)=t^3-2t^2+t,\ H^3_2(t)=t^3-t^2,\ H^3_3(t)=-2t^3+3t^2$

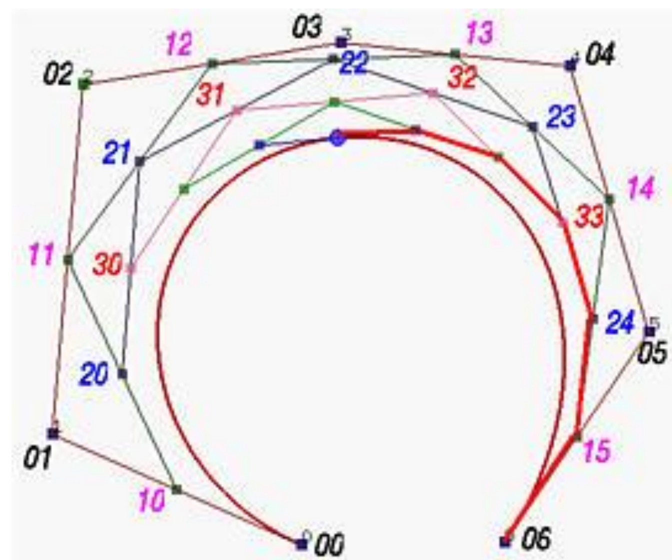
Bezier curve

- Approximation curve – mimicking shape of control polyline
- First and last control points (V_0, V_n) are interpolated
- $n.(V_1 - V_0)$, $n.(V_n - V_{n-1})$ are tangent vectors in V_0, V_n
- De Casteljau algorithm
  - Recursively computing point on curve for parameter t
  - $V^0_i(t) = V_i$, I = 0,…,n
  - $V^j_i(t) = (1-t)V^{j-1}_i(t) + tV^{j-1}_{i+1}(t)$, i=0,…,n-j, j=1,…n,
  - $B^n(t) = V^n_0(t)$
  - $V^{n-1}_1(t) - V^{n-1}_0(t)$ is tangent vector at $B^n(t)$
  - Decomposing curve to 2 Bezier curves, subdivision algorithm
    - $V^0_0(t), V^1_0(t) V^2_0(t), \ldots, V^n_0(t)$
    - $V^n_0(t), V^{n-1}_1, V^{n-2}_2(t) \ldots, V^0_n(t)$

Bezier curve



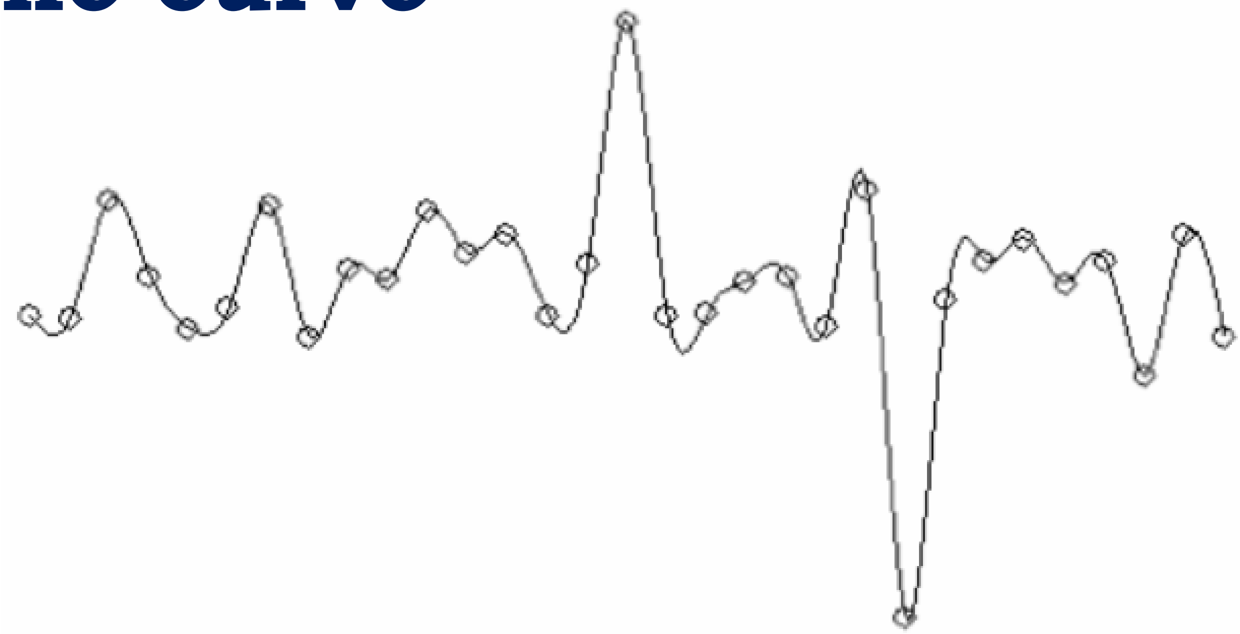


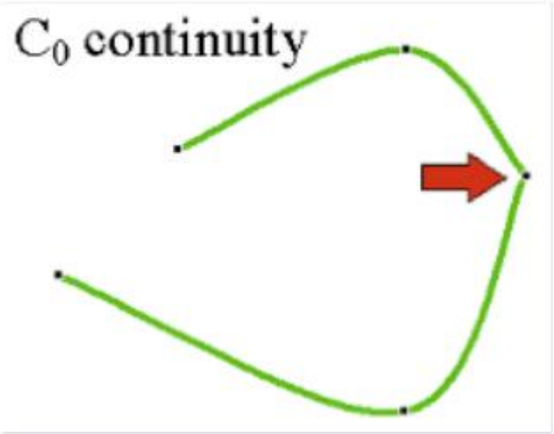


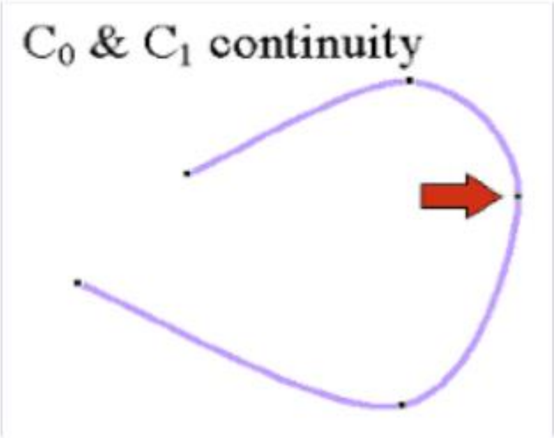
Spline curve

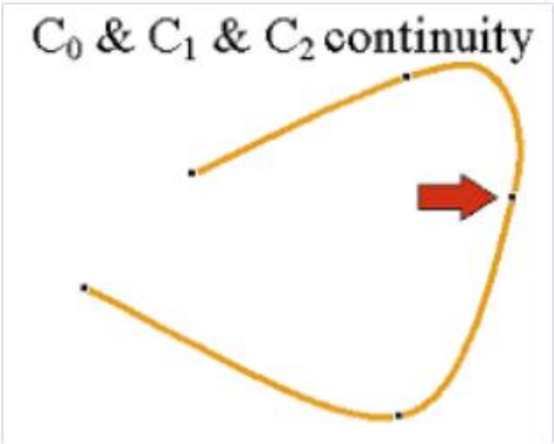
- Simple polynomial curve & many control points = high order of polynomials = slow computation
- Sticking together polynomial curves of small order - piecewise polynomial curve, consists of polynomial segments, segments meet at knots
- Representing each segment separately vs whole spline curve representation
- Expecting order of continuity at knots
  - C^0 – end point of first segment is equal to start point of second
  - C^1 – tangent vector at end point of first segment is equal to tangent vector at start point of second segment
  - G^1 – tangent vector at end point of first segment is multiplication of tangent vector at start point of second segment

Spline curve



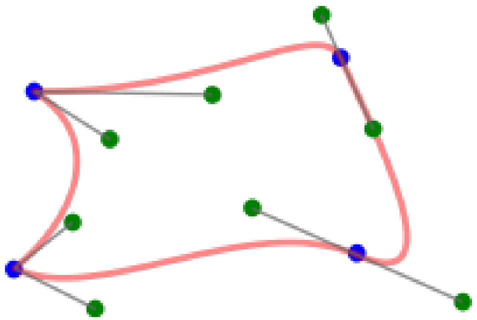


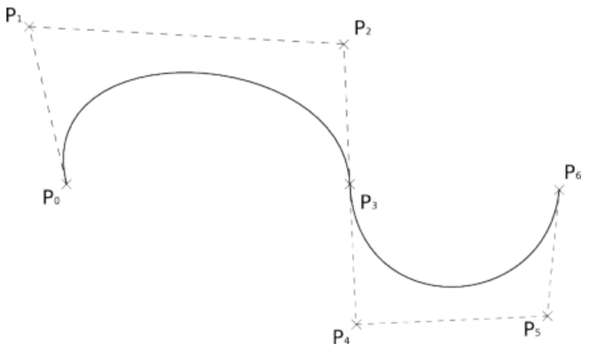




Bezier spline curve

- Each segment is represented as Bezier curve
- Usually linear, quadratic or cubic segments
- C^0 continuous Bezier spline – polybezier, beziergon
- C^1 continuous Bezier cubic spline
  - Given vertices $V_0, V_1, V_2, \ldots, V_n$, $n=3k$
  - $V_0, V_3, V_6, \ldots, V_{3k}$ – interpolated vertices
  - $V_{3k}=0.5V_{3k-1}+0.5V_{3k+1}$
- Used in PostScript, PDF, .ttf, OpenType, SVG, …





Hermite cubic spline curve

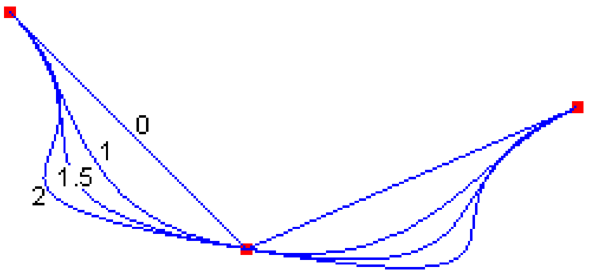
- Given vertex points $V_0, V_1, \ldots, V_n$, tangent vectors $T_0, T_1, \ldots, T_n$ and knot parameters $t_0 < t_1 < \ldots < t_n$
- Interpolation curve, interpolating each given vertex V_i and maintaining T_i as tangent vector at V_i
- Interpolation of tangents - C^1 continuity
- Used mainly for animation curves
- Each segment is polynomial and represented in Hermite cubic curve form
  - For $t \in \langle t_0, t_n \rangle$, pick span j such that $t \in \langle t_j, t_{j+1} \rangle$
  - $s = (t - t_j)/(t_{j+1} - t_j)$
  - $H(t) = S_j(s) = V_j H^3_0(s) + T_j H^3_1(s) + T_{j+1} H^3_j(s) + V_{j+1} H^3_3(s)$

Hermite cubic spline curve

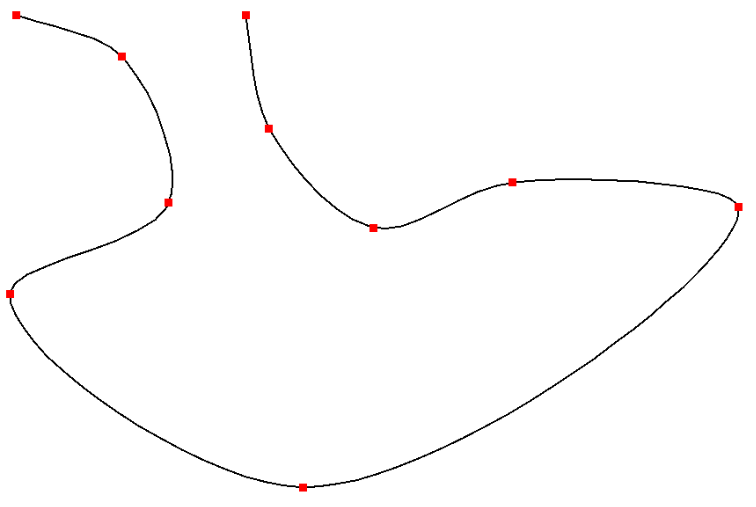
- Automatic computation of tangent vectors from given points and knot parameters
- Finite difference
  - $T_k = 0.5\left(\frac{V_{k+1}-V_k}{t_{k+1}-t_k} - \frac{V_k-V_{k-1}}{t_k-t_{k-1}}\right)$
- Cardinal spline
  - $T_k = (1-c)\frac{V_{k+1}-V_{k-1}}{t_{k+1}-t_{k-1}}$
  - c - tension
- Catmull-Rom spline
  - $T_k = \frac{V_{k+1}-V_{k-1}}{t_{k+1}-t_{k-1}}$
- Kochanek-Bartels spline
  - $T_k = \frac{(1-t)(1+b)(1+c)}{2}(T_k - T_{k-1}) + \frac{(1-t)(1-b)(1-c)}{2}(T_{k+1} - T_k)$
  - c – continuity, b – bias, t – tension

Hermite cubic spline curve

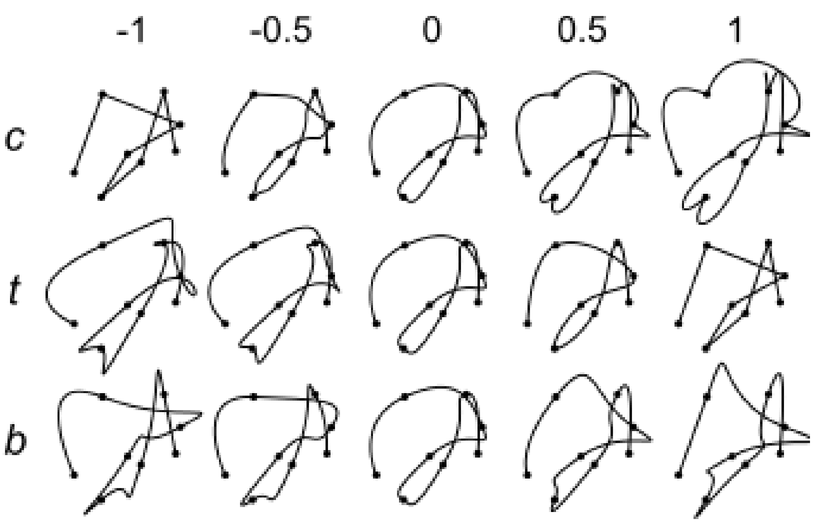
- Computation of knot parameters
  - Uniform: $t_k = k$
  - Length: $t_0 = 0$, $t_k = t_{k-1} + |V_k - V_{k-1}|$



Cardinal spline



Finite difference spline

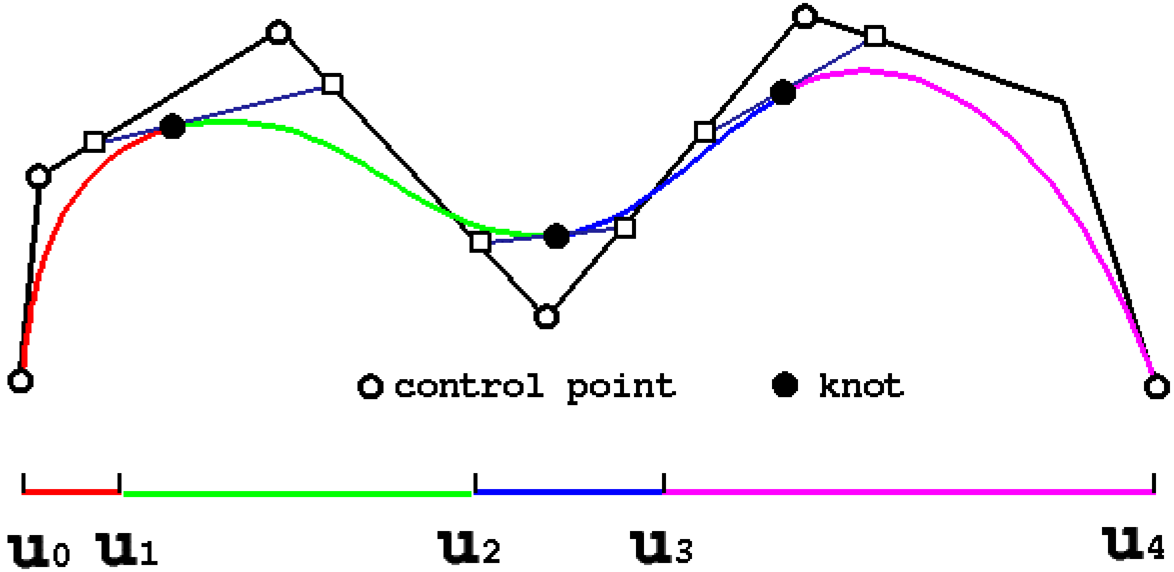


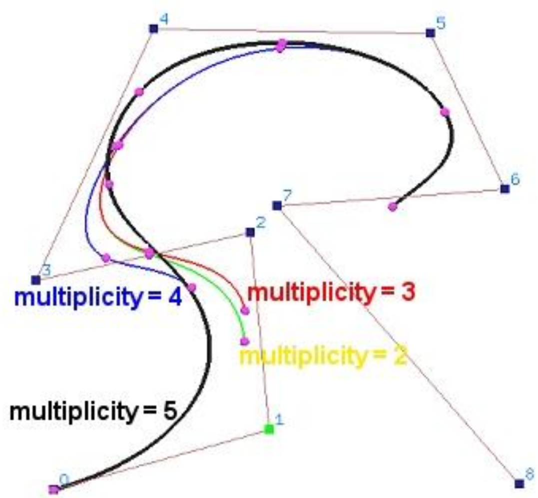
Kochanek-Bartels spline

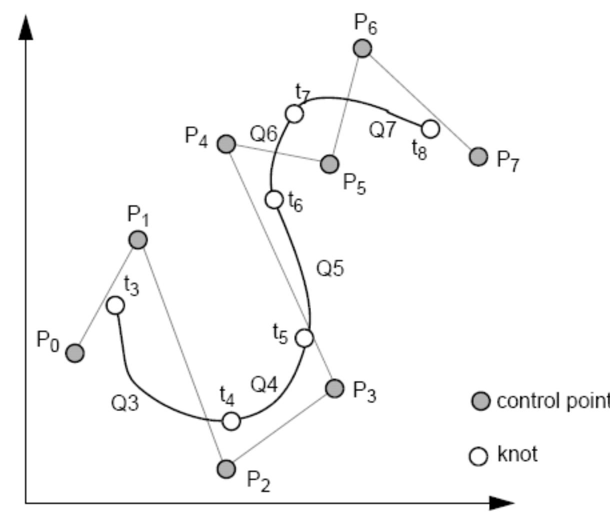
B-spline curve

- Compact representation of approximating spline curves
- Input
  - Polynomials degree d
  - Control points $V_0, V_1, \ldots, V_n$
  - Vector of knot parameters $t_0, t_1, \ldots, t_m$, $m=n+d+1$
- Knot vector represents polynomial segments (non-empty intervals in domain interval) and also order of continuity between segments (multiplicity of knot parameters)
- $BS^d(t) = \sum_{i=0}^{n} V_i N^d{}_i(t) \qquad t \in < t_d, t_{n+1})$
- B-spline basis functions
  - $N^0{}_i(t) = 1, t \in < t_i, t_{i+1})$
  - $N^0{}_i(t) = 0, t \notin < t_i, t_{i+1})$
  - $N^k{}_i(t) = \frac{t - t_i}{t_{i+k} - t_i} N^{k-1}{}_i(t) + \frac{t_{i+k+1} - t}{t_{i+k+!} - t_{i+1}} N^{k-1}{}_{i+1}(t)$
    - $i = 0, 1, \ldots, m-k-1 \qquad k = 1, 2, \ldots, d$
    - If some denominator is zero, whole fraction is equal to zero

B-spline curve







B-spline curve

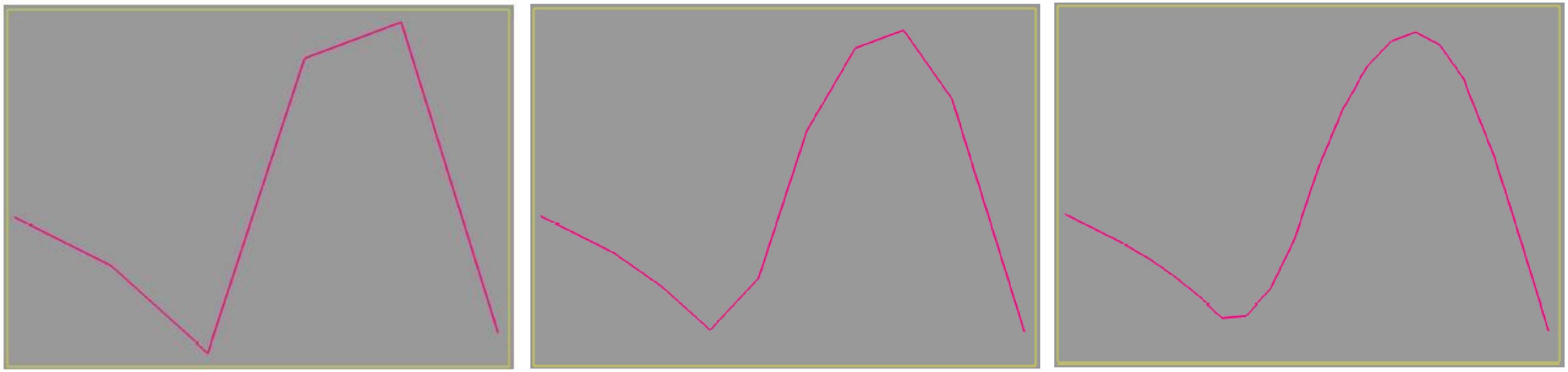
- If $t_0=t_1=\ldots=t_d$, curve starts at V_0
- If $t_{n+1}=t_{n+2}=\ldots=t_m$, limit of curve end is V_n
- Each segment is polynomial of maximal degree d
- If some knot parameter tj from domain has multiplicity q, then spline curve is C^{d-q} at that knot
- Number of polynomial segments is equal to number of different knot parameters in domain
- If each knot parameter has multiplicity d+1, control points are also control points of Bezier spline curve
- Local control – change of one control vertex affects only d segments in close vicinity of changed vertex
- Convex hull – whole curve lies in convex hull of its control points

B-spline curve

- http://web.mit.edu/hyperbook/Patrikalakis-Maekawa-Cho/node18.html
- De Boor evaluation algorithm
  - Recursive algorithm for curve point evaluation
  - Fast and numerically stable
  - Similar to de Casteljau algorithm
- Boehm knot insertion algorithm
  - Inserts one knot parameter into knot vector, refining knot vector and control points
  - Curve remains same, but its representation changes
- Knot removal algorithm
  - Removes one knot parameter from knot vector
  - Refines control points
  - Can change shape of curve

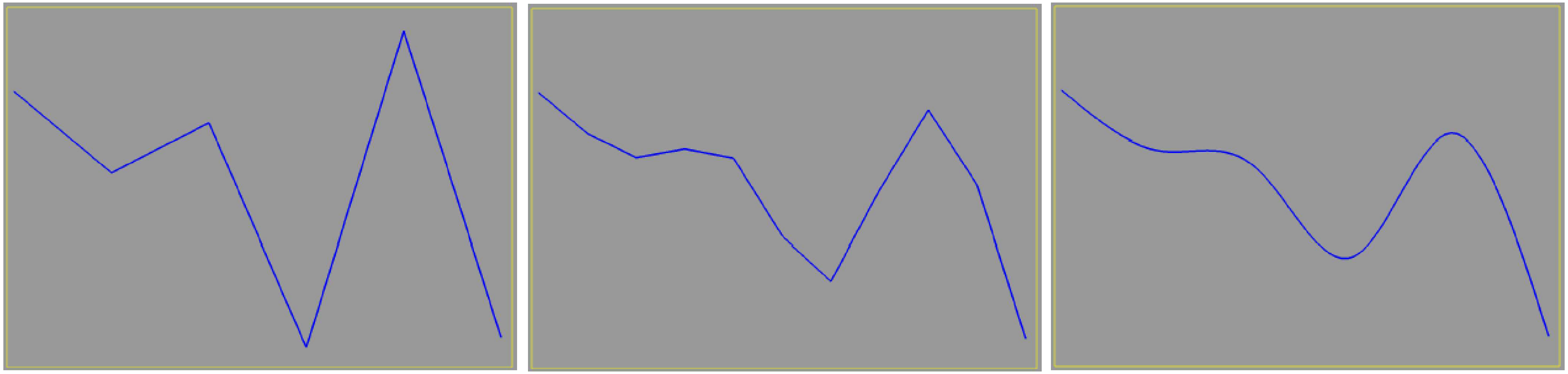
B-spline curve

- Define quadratic uniform B-spline curve, d=2
- Having control polygon $V_0, V_1, \ldots, V_n$
- Using uniform knot vector $0, 1, 2, \ldots, n+d+1$
- At one step, insert one knot into middle of each non-empty domain interval in knot vector
- Knot insertion algorithm defines Chaikin subdivision scheme for control polygon



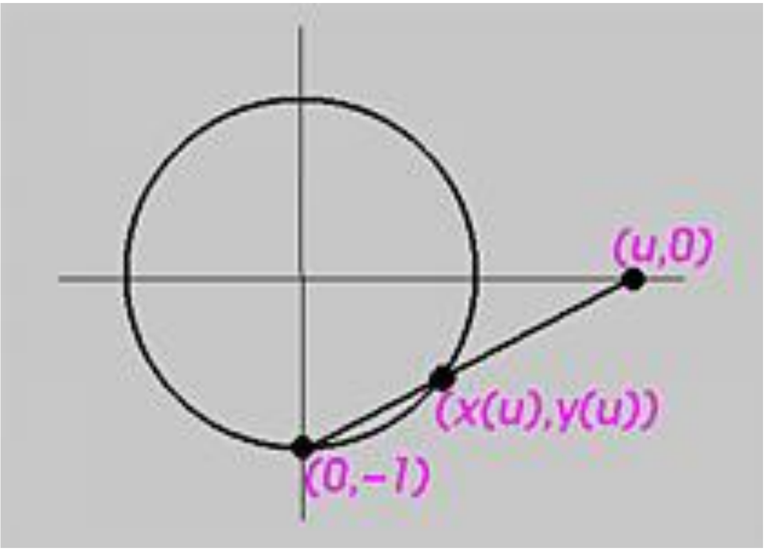
B-spline curve

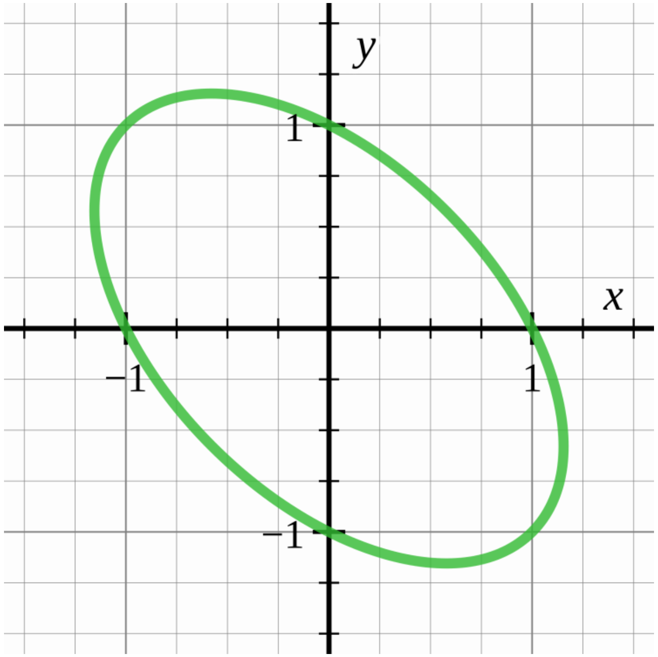
- Define cubic uniform B-spline curve, d=3
- Having control polygon $V_0, V_1, \ldots, V_n$
- Using uniform knot vector $0, 1, 2, \ldots, n+d+1$
- At one step, insert one knot into middle of each non-empty domain interval in knot vector
- Knot insertion algorithm defines Catmull-Clark subdivision scheme for control polygon



Rational curves

- Curve or its segments are made of rational functions
- Expanding class of representable curves
- Representation of conic sections
- Originated from projection of curve

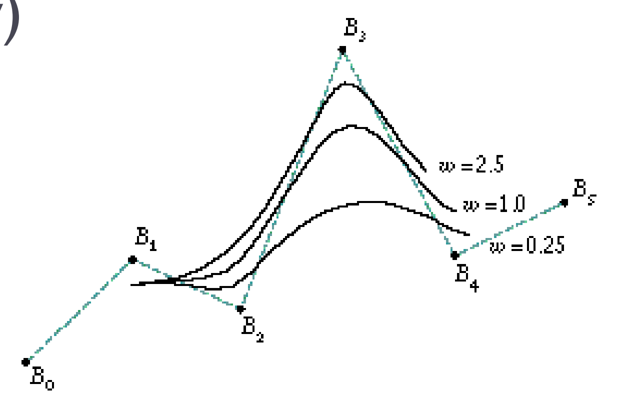




NURBS

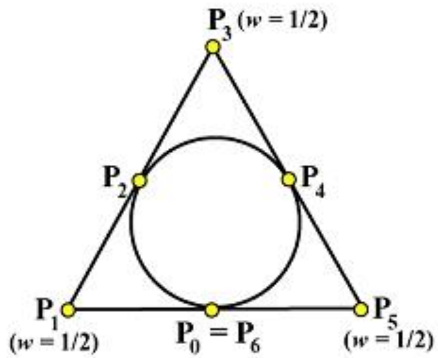
- Non-Uniform Rational B-spline
- Defining weights (real numbers) for each control point
- Embedding curve into space with additional dimension – into projective, homogenous space
  - $V_i=(x_i, y_i, z_i), w_i \rightarrow PV_i=(w_i x_i, w_i y_i, w_i z_i, w_i)$
- Evaluation, algorithms in projective space
- Projection of result point back to affine space
  - $PX=(x, y, z, w) \rightarrow X=(x/w, y/w, z/w)$

$$S(t)=\frac{\sum_{i=0}^{n} w_i V_i N_i^d(t)}{\sum_{i=0}^{n} w_i N_i^d(t)} \qquad t \in < t_d, t_{n+1} >$$

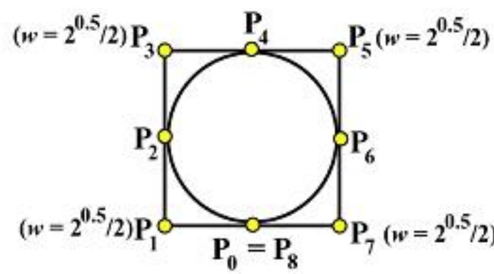


Conic sections

- Representing conic sections
- Circle as quadratic NURBS curve

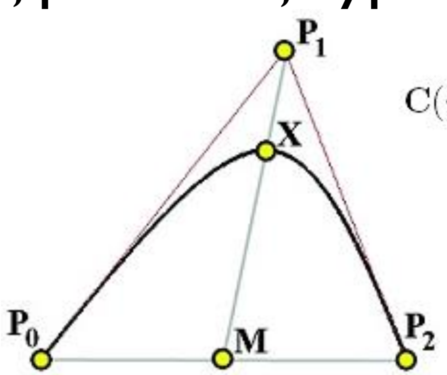


knotvector=[0,0,0,1,1,2,2,3,3,3]



knotvector=[0,0,0,1,1,2,2,3,3,4,4,4]

- Ellipse, parabola, hyperbola segments as rational Bezier curve

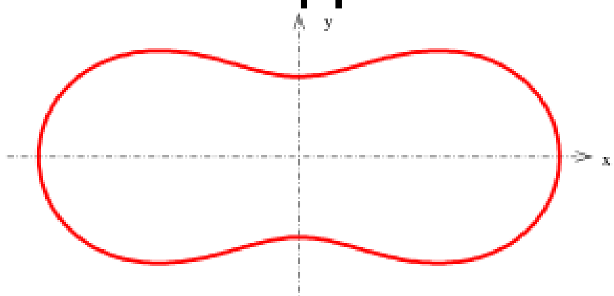


$$C(u) = \frac{1}{(1-u)^2 + 2(1-u)uw + u^2}\left((1-u)^2 P_0 + 2(1-u)uwP_1 + u^2 P_2\right)$$

w=1	parabola
w<1	ellipse
w>1	hyperbola

Implicit curve

- Algebraic curves
- 2D: Set of all points X ϵ E2 such that f(X)=0
  - Circle: $x^2+y^2-r^2=0$
- 3D: Set of all points X ϵ E3 such that f(X)=0, g(X)=0
  - Circle: $x^2+y^2+z^2-r^2=0$, $x+y+z=0$
- Easy computation if some point is on curve
- Defining interior, exterior regions by sign of f
- Hard to generate points on curve – hard visualization
- Used for smooth approximation of geometric objects



$$(x^2+y^2)^2-2c^2(x^2-y^2)-(a^4-c^4)=0$$

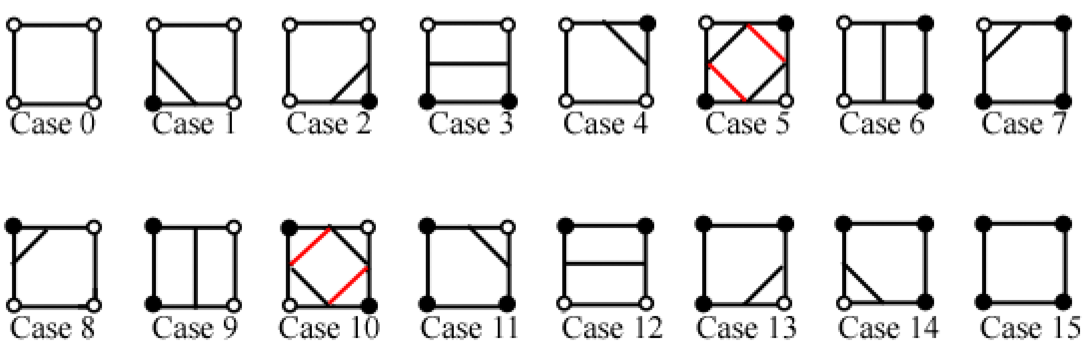
a=1.1
c=1

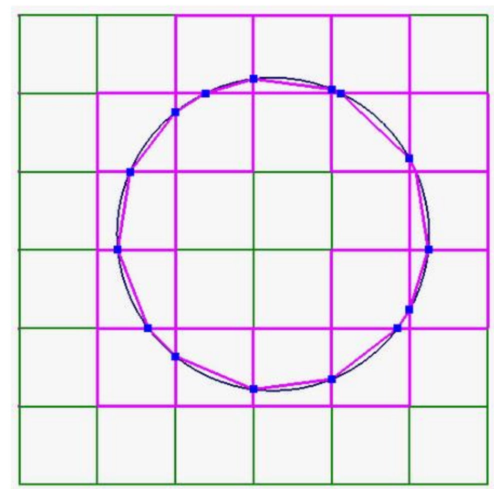
Implicit curve

- Visualization algorithms
- Points generation
  - For space point $Q=(x_0,y_0)$, iteratively find point close enough to curve
  - Finding solution in the direction of gradient (first derivation)
  - Newton method for solving $f(Q+t.(f_x(Q),f_y(Q)))=0$
  - $(x_{i+1}, y_{i+1}) = (x_i, y_i) - \frac{f(x_i,y_i)}{f_x(x_i,y_i)^2 + f_y(x_i,y_i)^2}(f_x(x_i, y_i), f_y(x_i, y_i))$
  - Finish iteration when change after one step is small
- Tracing algorithm
  - Find starting point near curve Q_1
  - Determine point P_1 from Q_1 using Newton method
  - Determine tangent vector T_1 in P_1 and compute $Q_2=P_1+sT_1$ (s-step)
  - Repeat until we are back in P_1
  - Polyline $P_1,P_2,\ldots,P_n$ is approximation of implicit curve

Implicit curve

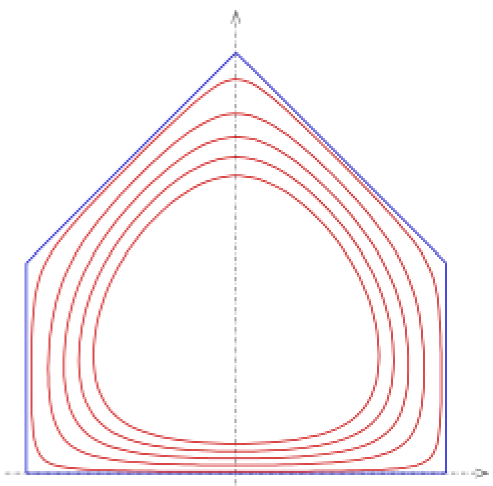
- Visualization algorithms
- Marching squares
  - Divide space using uniform grid
  - For each grid point, compute value of f
  - For each cell in grid, generate line segments based on values of f in cell's corners
  - Using linear interpolation to compute end points of segments
  - Render generated line segments

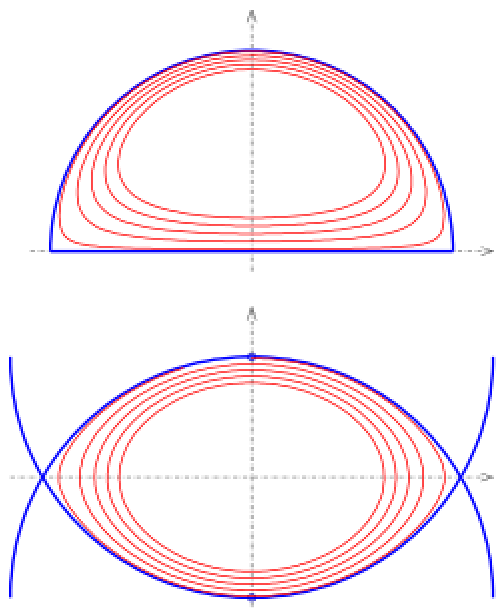


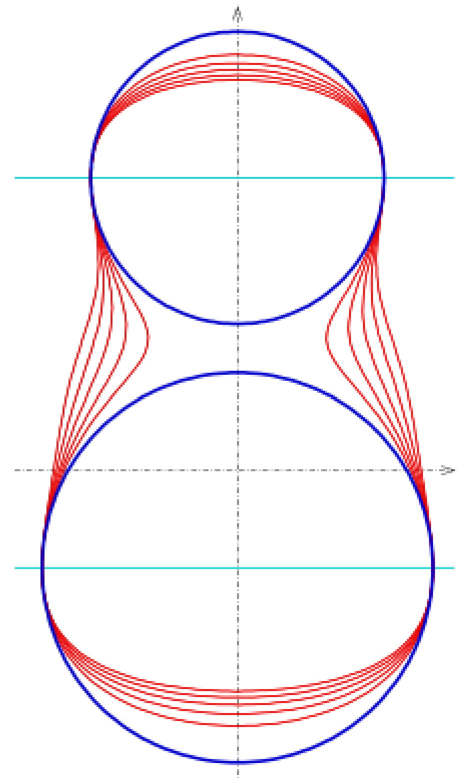


Implicit curve

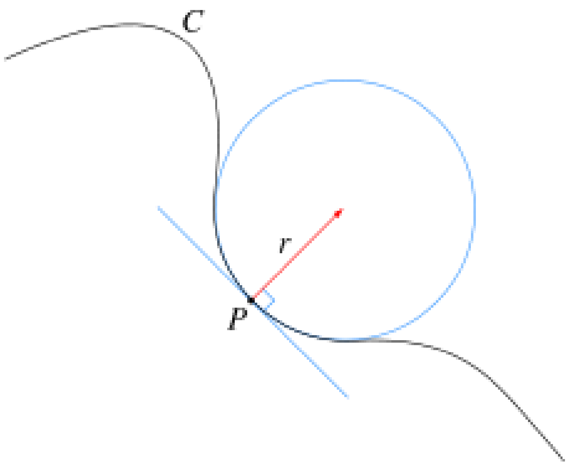
- Approximation of blending, intersection
- $f(X)=g_1(X).g_2(X)\ldots g_n(X)-c$







Differential geometry



- Parametric curve
  - Tangent vector – $T = \frac{\partial f(t)}{\partial t}$
  - Normal vector – $N = \frac{\partial^2 f(t)}{\partial t^2}$
  - Curvature – fitting best circle at point
  - Curvature - $k = \frac{\left|\frac{\partial f(t)}{\partial t} \times \frac{\partial^2 f(t)}{\partial t^2}\right|}{\left|\frac{\partial f(t)}{\partial t}\right|^3}$
- Implicit curve
  - Gradient, normal vector - $\nabla f = N = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right) = (f_x, f_y)$
  - Curve is regular at point if gradient is not zero vector
  - Tangent vector - $T = (-\frac{\partial f}{\partial y}, \frac{\partial f}{\partial x})$
  - Curvature - $k = \frac{-{f_y}^2 f_{xx} + 2 f_x f_y f_{xy} - {f_x}^2 f_{yy}}{({f_x}^2 + {f_y}^2)^{1,5}}$

The End
for today