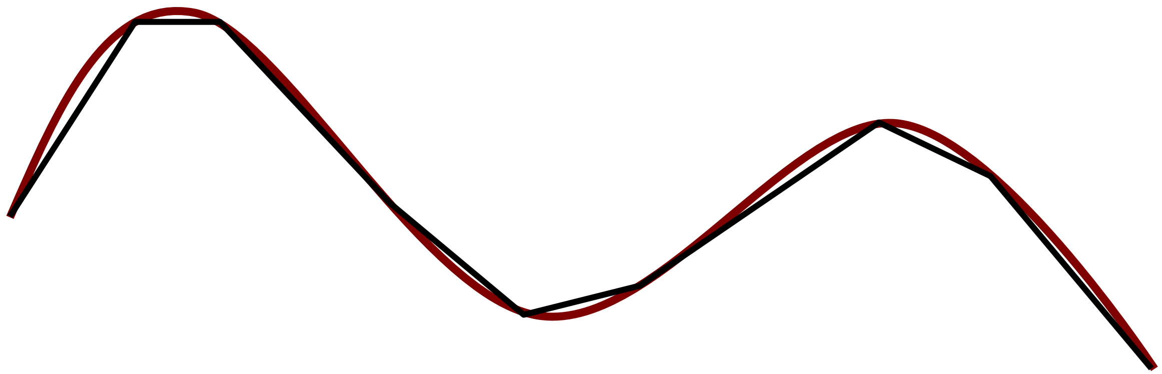
CENG 477
Introduction to Computer Graphics

Representing Curves and Surfaces

Introduction

- There are no perfectly straight lines or flat faces in nature!
- Therefore, representing and generating smooth shapes is a requirement in many CG applications
- Rendering will still use lines and triangles but their vertices will be sampled from a curve or surface



Curves

- There are many ways to represent curves:
  - must be practical (easy to manage and render)
  - must be flexible (general enough to be used in various modeling tasks)
- A good compromise is cubic polynomials
- Each x, y, z coordinate is expressed as a cubic polynomial with potentially different coefficients (t is the parameter)

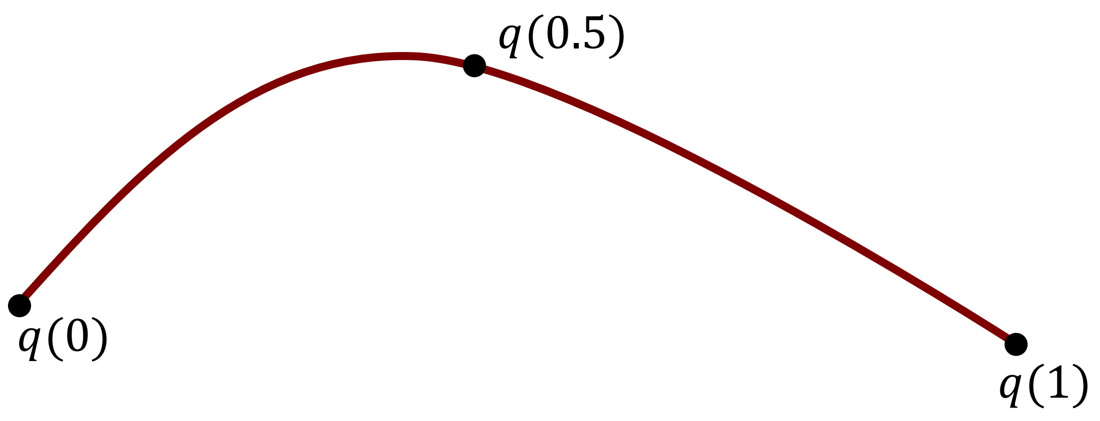
$$x(t) = a_x t^3 + b_x t^2 + c_x t + d_x$$

$$y(t) = a_y t^3 + b_y t^2 + c_y t + d_y$$

$$z(t) = a_z t^3 + b_z t^2 + c_z t + d_z$$

Curves

- For a given t value, $q(t) = [x(t)\ y(t)\ z(t)]$represents the 3D position along the curve
- Similar to rays in ray tracing except that it may follow a curvy path instead of a straight one!
- The t parameter is taken to be in range $[0, 1]$



Cubic Polynomials

- As cubic polynomials have 4 unknowns (per component), we need 4 constraints to find them
- Different curves are distinguished by different constraints
  - **Hermite curves:** 2 end points + 2 tangent vectors
  - **Bezier curves:** 2 end points + 2 control points
  - **Splines:** 4 control points (for each piece of the curve)

Matrix Form

- Cubic polynomials are conveniently expressed in matrix form:

$$x(t) = a_x t^3 + b_x t^2 + c_x t + d_x$$

$$y(t) = a_y t^3 + b_y t^2 + c_y t + d_y$$

$$z(t) = a_z t^3 + b_z t^2 + c_z t + d_z$$

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

$$Q(t) = TC$$

Matrix Form

- We will also need the derivative of this curve to specify tangent vectors

$$\frac{dQ(t)}{dt} = Q'(t) = \frac{dT}{dt}C = [3t^2 \ 2t \ 1 \ 0]C$$

Constraints

- Imagine that we want to specify certain geometrical constraints such as:
  - Start point
  - End point
  - Start direction (i.e. tangent vector at start point)
  - End direction (i.e. tangent vector at end point)
- We need to split the matrix C into two to allow embedding of these constraints

Constraints

- Rewrite $C = MG$, where G represents the geometry constraints

$$C = \begin{bmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \\ d_x & d_y & d_z \end{bmatrix} = \underbrace{\begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ m_{41} & m_{42} & m_{43} & m_{44} \end{bmatrix}}_{M} \underbrace{\begin{bmatrix} g_{1x} & g_{1y} & g_{1z} \\ g_{2x} & g_{2y} & g_{2z} \\ g_{3x} & g_{3y} & g_{3z} \\ g_{4x} & g_{4y} & g_{4z} \end{bmatrix}}_{G}$$

- Here, M is called the basis matrix
- G is called the geometry or the constraints matrix
- Different types of curves differ in M and G

Constraints

- Note that $Q(t) = TMG$
- For instance:

$$x(t) = [t^3 \; t^2 \; t \; 1]\left(g_{1x}\begin{bmatrix} m_{11} \\ m_{21} \\ m_{31} \\ m_{41} \end{bmatrix} + g_{2x}\begin{bmatrix} m_{12} \\ m_{22} \\ m_{32} \\ m_{42} \end{bmatrix} + g_{3x}\begin{bmatrix} m_{13} \\ m_{23} \\ m_{33} \\ m_{43} \end{bmatrix} + g_{4x}\begin{bmatrix} m_{14} \\ m_{24} \\ m_{34} \\ m_{44} \end{bmatrix} \right)$$

Blending Functions

- Rewriting this gives us:

$$x(t) = [t^3\ t^2\ t\ 1]\left(g_{1x}\begin{bmatrix} m_{11} \\ m_{21} \\ m_{31} \\ m_{41} \end{bmatrix} + g_{2x}\begin{bmatrix} m_{12} \\ m_{22} \\ m_{32} \\ m_{42} \end{bmatrix} + g_{3x}\begin{bmatrix} m_{13} \\ m_{23} \\ m_{33} \\ m_{43} \end{bmatrix} + g_{4x}\begin{bmatrix} m_{14} \\ m_{24} \\ m_{34} \\ m_{44} \end{bmatrix} \right)$$

$$\Downarrow$$

$$\begin{aligned} x(t) = {} & (t^3 m_{11} + t^2 m_{21} + t m_{31} + m_{41}) g_{1x} + \\ & (t^3 m_{12} + t^2 m_{22} + t m_{32} + m_{42}) g_{2x} + \\ & (t^3 m_{13} + t^2 m_{23} + t m_{33} + m_{43}) g_{3x} + \\ & (t^3 m_{14} + t^2 m_{24} + t m_{34} + m_{44}) g_{4x} \end{aligned}$$

- That is, the curve is a weighted sum of the elements of the geometry matrix

Blending Functions

- The weights are each cubic polynomials of t
- These polynomials are called blending functions

$$\begin{aligned} x(t) = {} & (t^3 m_{11} + t^2 m_{21} + t m_{31} + m_{41}) g_{1x} + \\ & (t^3 m_{12} + t^2 m_{22} + t m_{32} + m_{42}) g_{2x} + \\ & (t^3 m_{13} + t^2 m_{23} + t m_{33} + m_{43}) g_{3x} + \\ & (t^3 m_{14} + t^2 m_{24} + t m_{34} + m_{44}) g_{4x} \end{aligned}$$

$$\Downarrow$$

$$B_{1x4} = T_{1x4} M_{4x4}$$

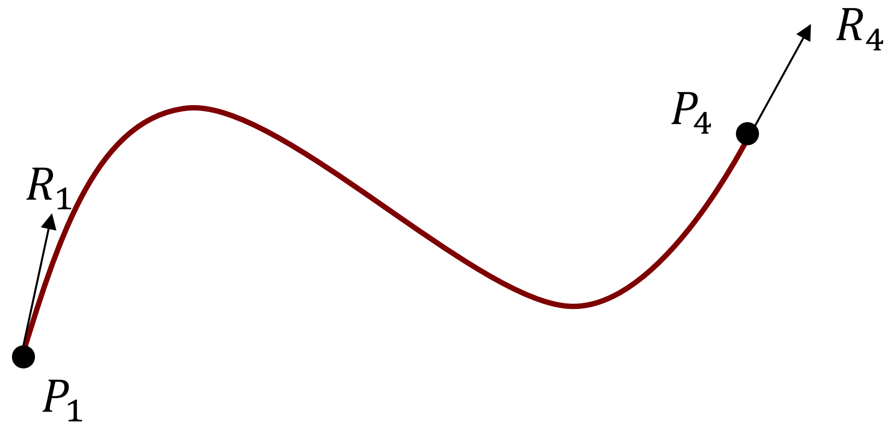
Vector of
blending functions

Two Important Curves

- Now we will see how this background can be used to define two types of curves:
  - Hermine curves
  - Bezier curves
- Both curves can draw the same curves (they are equally powerful) but they have different geometry constraints

Hermite Curves

- The constraints of Hermite curves are:
  - **Two end points:** P_1 and P_4
  - **Two tangent vectors:** R_1 and R_4



Hermite Curves

- The geometry matrix then becomes:

$$G = \begin{bmatrix} P_1 \\ P_4 \\ R_1 \\ R_4 \end{bmatrix} = \begin{bmatrix} P_{1x} & P_{1y} & P_{1z} \\ P_{4x} & P_{4y} & P_{4z} \\ R_{1x} & R_{1y} & R_{1z} \\ R_{4x} & R_{4y} & R_{4z} \end{bmatrix}$$

- To find M remember that the curve in matrix form we have:

$$Q(t) = TMG$$

- And for derivative: $Q'(t) = T'MG$

Hermite Curves

- We can now plug in values for the t parameter
  - Compute $Q(0), Q(1), Q'(0), Q'(1)$

$$Q(0) = [0\ 0\ \ 0\ 1]MG$$
$$Q(1) = [1\ 1\ \ 1\ 1]MG$$
$$Q'(0) = [0\ 0\ \ 1\ 0]MG$$
$$Q'(1) = [3\ 2\ \ 1\ 0]MG$$

- This is the same as:

$$\begin{bmatrix} Q(0) \\ Q(1) \\ Q'(0) \\ Q'(1) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} MG$$

Hermite Curves

- Remember that G was equal to:

$$G = \begin{bmatrix} P_1 \\ P_4 \\ R_1 \\ R_4 \end{bmatrix} = \begin{bmatrix} Q(0) \\ Q(1) \\ Q'(0) \\ Q'(1) \end{bmatrix}$$

- So we have:

$$\underbrace{\begin{bmatrix} Q(0) \\ Q(1) \\ Q'(0) \\ Q'(1) \end{bmatrix}}_{\text{Same}} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} M \underbrace{\begin{bmatrix} Q(0) \\ Q(1) \\ Q'(0) \\ Q'(1) \end{bmatrix}}_{\text{Same}}$$

Only possible
if M is the inverse
of the matrix

Hermite Curves

- Therefore, Hermite curves have the following basis matrix:

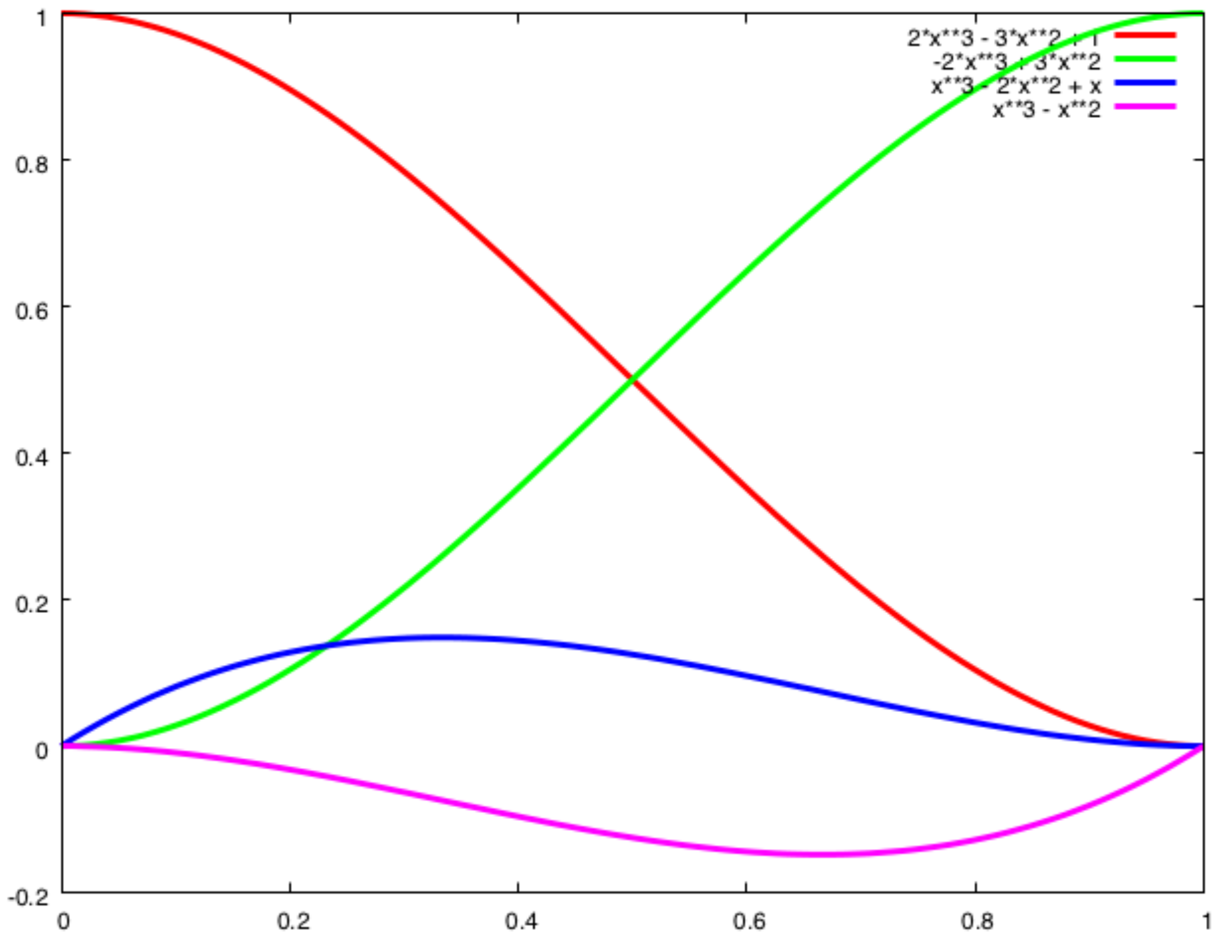
$$M = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix}^{-1} = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

- This yields the following blending functions:

$$B = \begin{bmatrix} 2t^3 - 3t^2 + 1 \\ -2t^3 + 3t^2 \\ t^3 - 2t^2 + t \\ t^3 - t^2 \end{bmatrix}^T$$

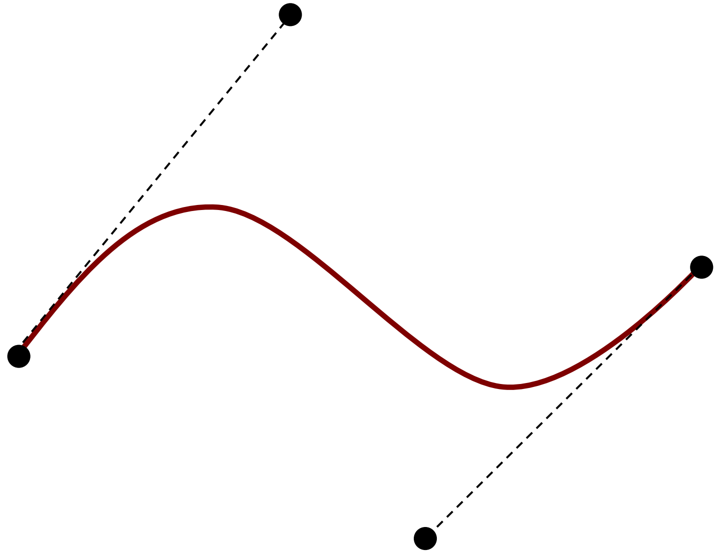
Let's plot these
with gnuplot and
do some experimentation
with Matlab!

Hermite Blending Functions



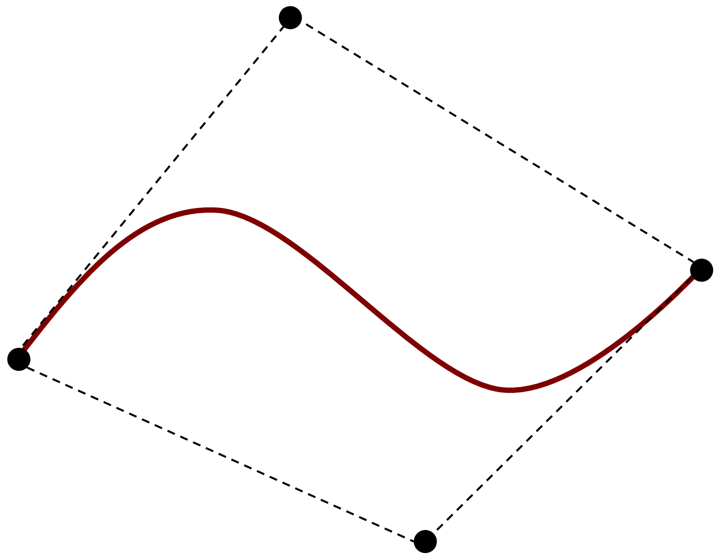
Bezier Curves

- Bezier curves can draw the same curves as Hermite curves
- They are defined using control points instead of derivatives
- Two control points are interpolated and two control points are approximated



Bezier Curves

- The curve lies entirely within the convex-hull of these four points
- This is useful for clipping, culling, and intersection tests as the convex-hull can be tested first instead of each line segment



Bezier Curves

- Bezier curves are related to the Hermite curves as:

$$R_1 = 3(P_2 - P_1)$$

$$R_4 = 3(P_4 - P_3)$$

- The factor 3 ensures that P_2 has the highest weight at $t = 1/3$ and P_3 has the highest weight at $t = 2/3$, logically dividing the curve into 3 pieces

Bezier Curves

- In matrix form, this relationship can be express as:

$$G_H = \begin{bmatrix} P_1 \\ P_4 \\ R_1 \\ R_4 \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ -3 & 3 & 0 & 0 \\ 0 & 0 & -3 & 3 \end{bmatrix}}_{\text{This matrix translates Bezier geometry matrix to the Hermite geometry matrix (let's call this matrix as } M_{BH}\text{)}} \underbrace{\begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{bmatrix}}_{G_B}$$

Bezier Curves

- Then Bezier curves can be defined as:

$$Q(t) = TM_H G_H = TM_H M_{BH} G_B$$

$$Q(t) = TM_B G_B$$

where $M_B = M_H M_{BH}$

$$Q(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{bmatrix}$$

Bezier Curves

- This is equivalent to:

$$
\begin{aligned}
Q(t) = \; & (-t^3 + 3t^2 - 3t + 1)P_1 + \\
& (3t^3 - 6t^2 + 3t)P_2 + \\
& (-3t^3 + 3t^2)P_3 + \\
& (t^3)P_4
\end{aligned}
\quad = \quad
\begin{aligned}
& (1-t)^3 P_1 + \\
& 3t(1-t)^2 P_2 + \\
& 3t^2(1-t)P_3 + \\
& t^3 P_4 +
\end{aligned}
$$

- These are called Bernstein polynomials
  - Their sum is always 1
  - They are always non-negative when $t \in [0,1]$
  - That's why the resulting curve is in the convex-hull of P_1, P_2, P_3, P_4
  - Bernstein polynomial of degree n is $B_{i,n}(t) = \binom{n}{i} t^i (1-t)^{n-i}$

Bezier Curves

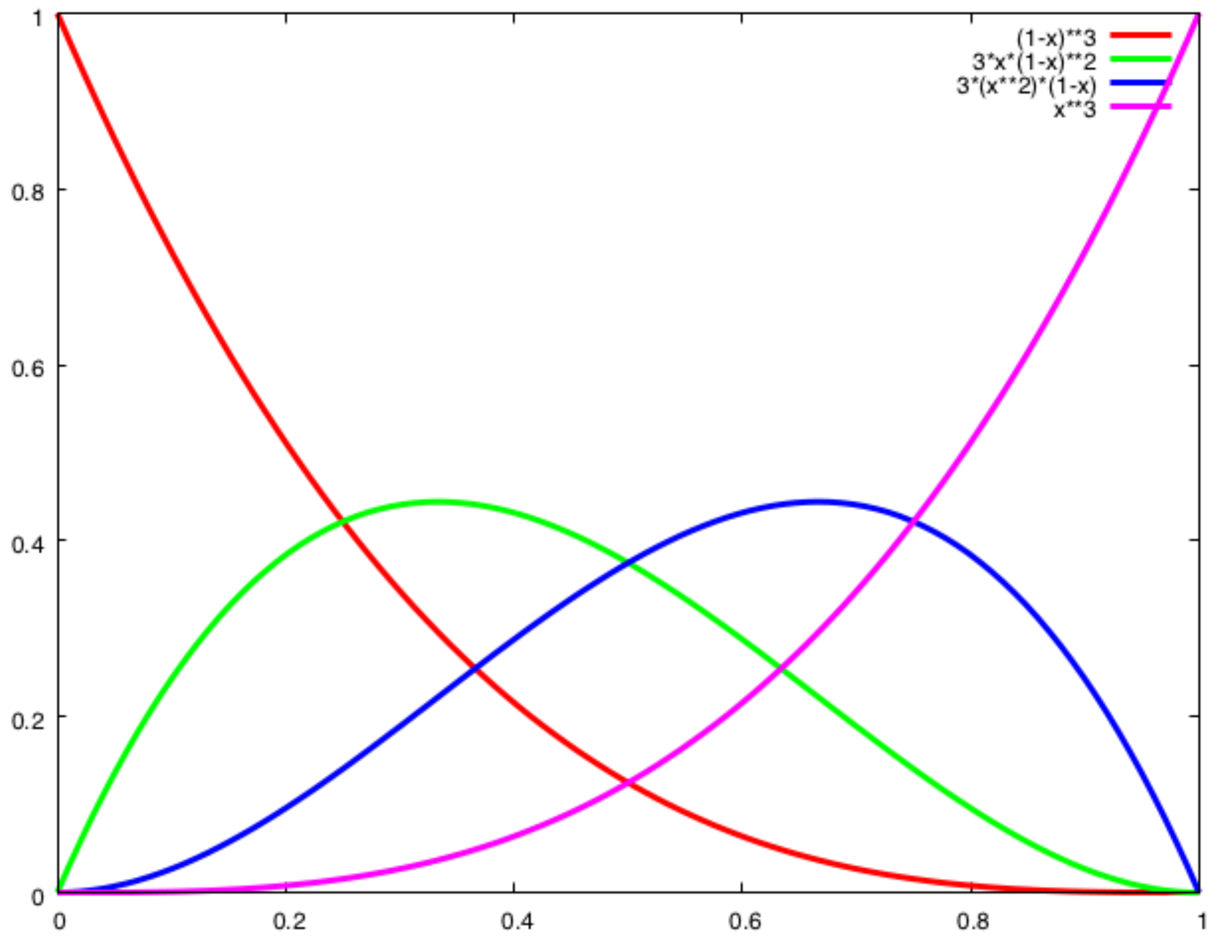
- Bezier curve is the sum of Bernstein polynomials

$$Q(t) = (1-t)^3 P_1 + 3t(1-t)^2 P_2 + 3t^2(1-t) P_3 + t^3 P_4$$

$$Q(t) = \sum_{i=0}^{n} B_{i,n}(t) P_{i+1}$$

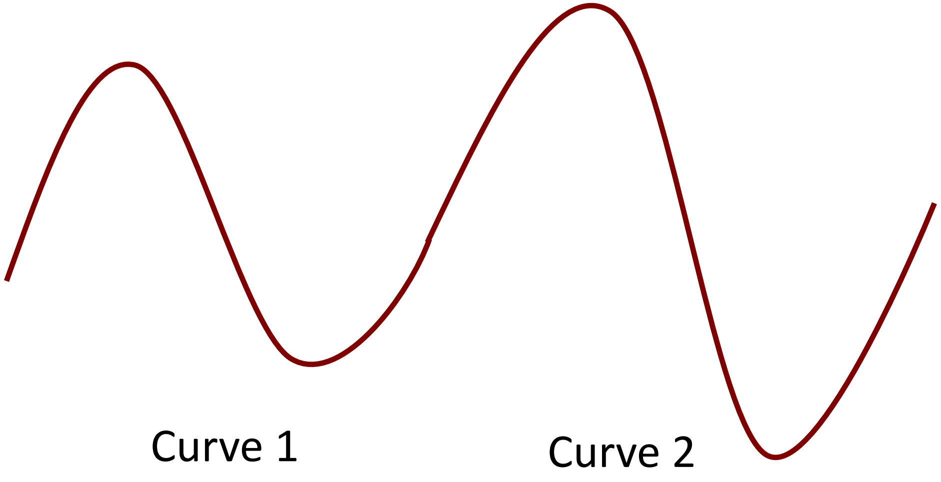
$$B_{i,n}(t) = \binom{n}{i} t^i (1-t)^{n-i}$$

Bernstein Polynomials



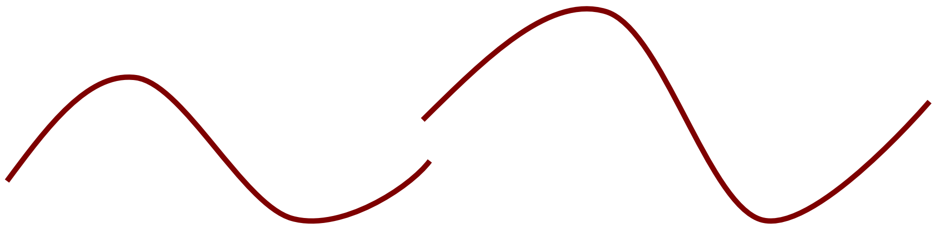
Continuity

- Until now, we learned to draw a single curve segment
- If we want to combine multiple curve segments, we must ensure maintaining continuity

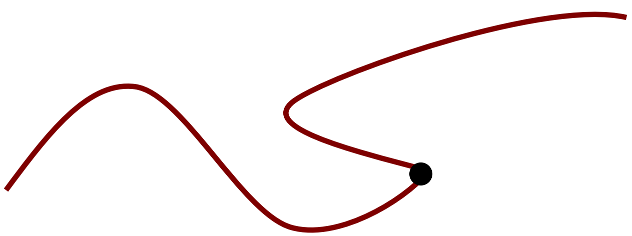


Types of Continuity

- **No continuity:**
  - The curves do not meet

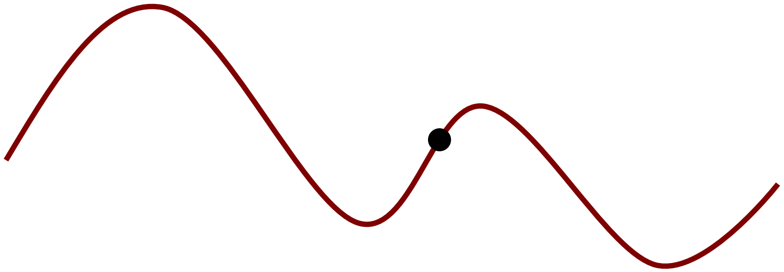


- **C0 continuity:**
  - The end points meet, also know as positional continuity

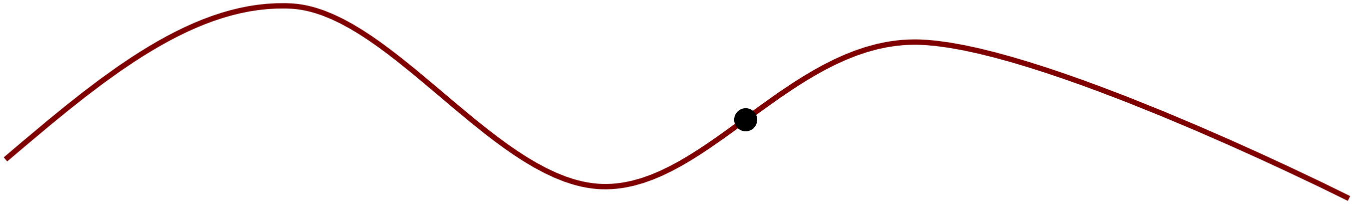


Types of Continuity

- **C1 continuity:**
  - The curves meet and have identical tangent vectors at the connection



- **C2 continuity:**
  - The curves meet and have identical curvature at the connection
  - The curvature is defined as the rate of change of tangents



Types of Continuity

- Imagine a camera moving along a curve with multiple segments
  - **No continuity:** camera will make jumps between segments
  - **C0 continuity:** camera velocity may suddenly change
  - **C1 continuity:** camera acceleration may suddenly change
  - **C2 continuity:** camera motion will appear smooth
- In general, maintaining C2 continuity is desired

Maintaining Continuity

- Imagine having two Hermite curves:

$$G_l = \begin{bmatrix} P_1 \\ P_4 \\ R_1 \\ R_4 \end{bmatrix} \qquad G_r = \begin{bmatrix} Q_1 \\ Q_4 \\ T_1 \\ T_4 \end{bmatrix}$$

- C1 continuity can be maintained if $P_4 = Q_1$ and $R_4 = T_1$

Maintaining Continuity

- Similarly for two Bezier curves:

$$G_l = \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \end{bmatrix} \qquad G_r = \begin{bmatrix} Q_1 \\ Q_2 \\ Q_3 \\ Q_4 \end{bmatrix}$$

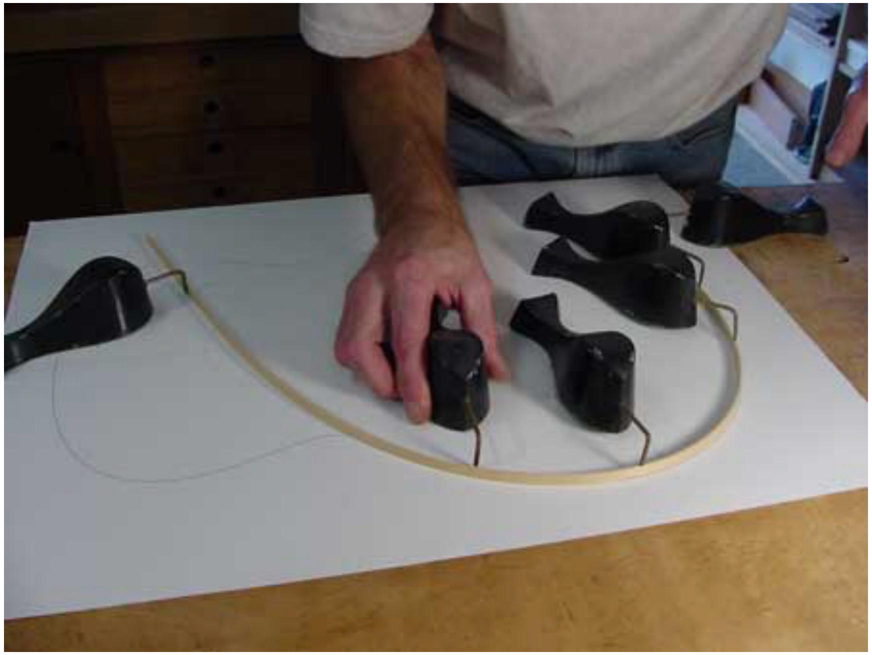
- C1 is ensured if $P_4 = Q_1$ and $P_4 - P_3 = Q_2 - Q_1$

Maintaining Continuity

- What if we want to maintain C2 continuity?
- Unfortunately, neither Hermite nor Bezier curves can guarantee C2 continuity
- For this we have a new type of curve called splines

Splines

- The term spline was used to refer to flexible metal strips used by draftspersons to design the surfaces of airplanes, cars, and ships, etc.



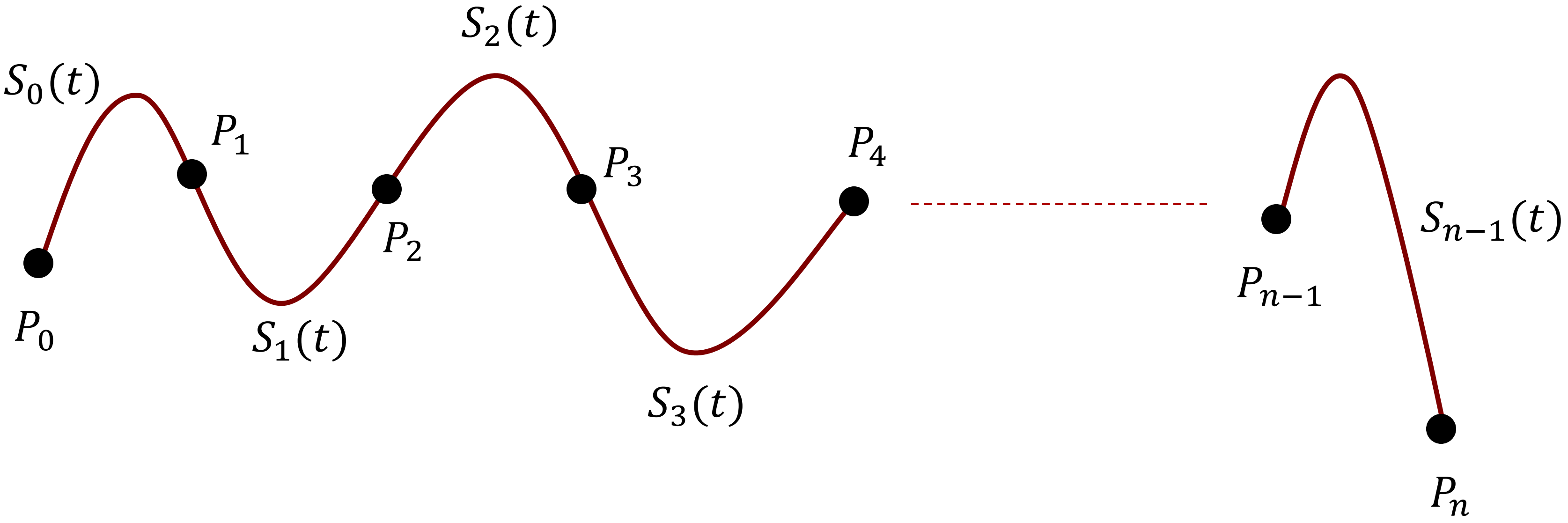
core77.com

Splines

- The splines, due to physical properties of the metal strips, had second order (C2) continuity
- Its mathematical equivalent is natural cubic splines
- Splines have one more degree of continuity than that is afforded by Hermite and Bezier curves
- There are other types of splines:
  - B-Splines
  - Uniform Nonrational B-Splines
  - Nonuniform Nonrational B-Splines
  - Nonuniform Rational B-Splines
  - Beta-Splines
  - V-Splines

Natural Cubic Splines

- Defined by *n+1* control points
- The spline, consisting of *n* curves, interpolates all of these points



Natural Cubic Splines

- The spline is defined as:

$$S(t) = \begin{cases} S_0(t), & t_0 \le t \le t_1 \\ S_1(t), & t_1 \le t \le t_2 \\ \vdots & \\ S_{n-1}(t), & t_{n-1} \le t \le t_n \end{cases}$$

- Each curve is a cubic polynomial:

$$S_0(t) = a_0 t^3 + b_0 t^2 + c_0 t + d_0$$

$$\vdots$$

$$S_{n-1}(t) = a_{n-1} t^3 + b_{n-1} t^2 + c_{n-1} t + d_{n-1}$$

There are a total of *4n* unknowns!

Natural Cubic Splines

- The end points of the curves must meet (C0 cont.) and their first two derivatives must be equal (C1 and C2):

$$\left.\begin{aligned} S_{i-1}(t_i) &= S_i(t_i) \\ S'_{i-1}(t_i) &= S'_i(t_i) \\ S''_{i-1}(t_i) &= S''_i(t_i) \end{aligned}\right\} \quad i = 1 \ldots n-1$$

- This gives us *3n – 3* equations

Natural Cubic Splines

- We also know the values of the spline at the control points:

$$S(t_i) = P_i, \qquad i = 0 \dots n$$

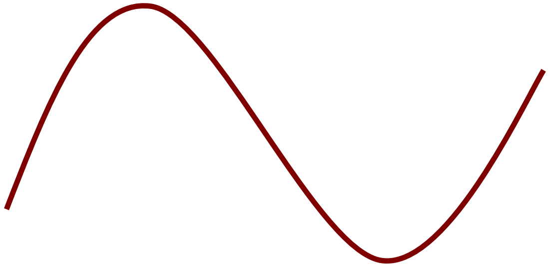
- This gives us another *n+1* equations
- We still need two more …
- $S''(t_0) = S''(t_n) = 0$ gives us natural cubic splines

Computing Polynomials

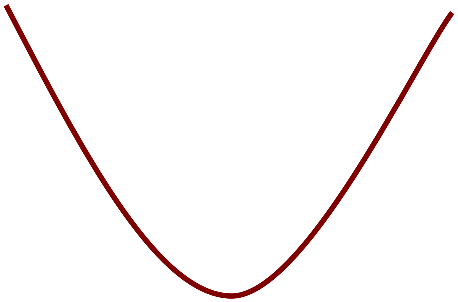
- Let's call the second derivatives at control points as:

$$Z_i = S''(t_i)$$

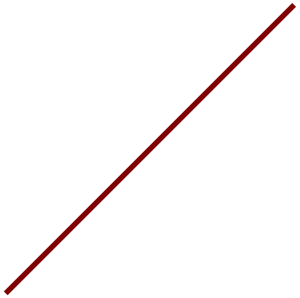
- For natural cubic spline we have $Z_0 = Z_n = [0\ 0\ 0]^T$
- How does $S''(t)$ look like?



$S(t)$



$S'(t)$

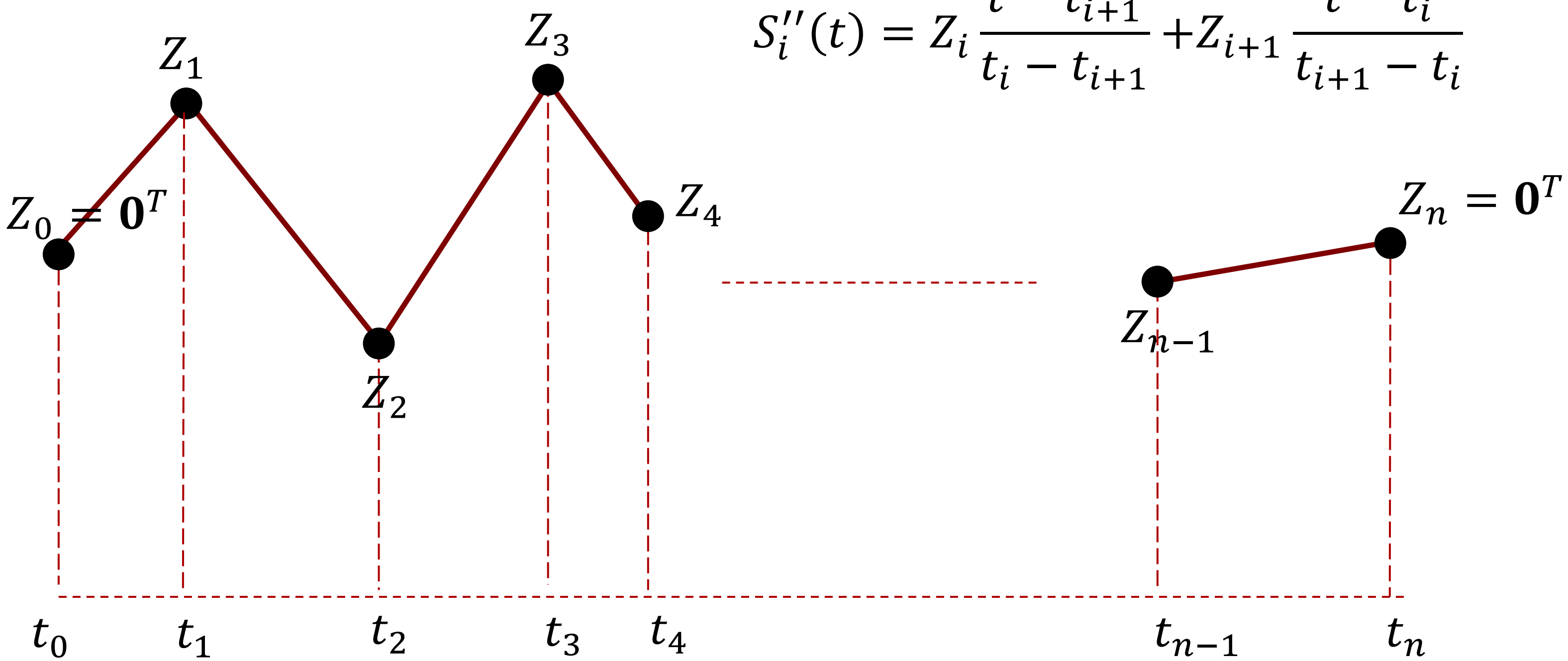


$S''(t)$

Computing Polynomials

- $S''(t)$ will be piecewise-linear

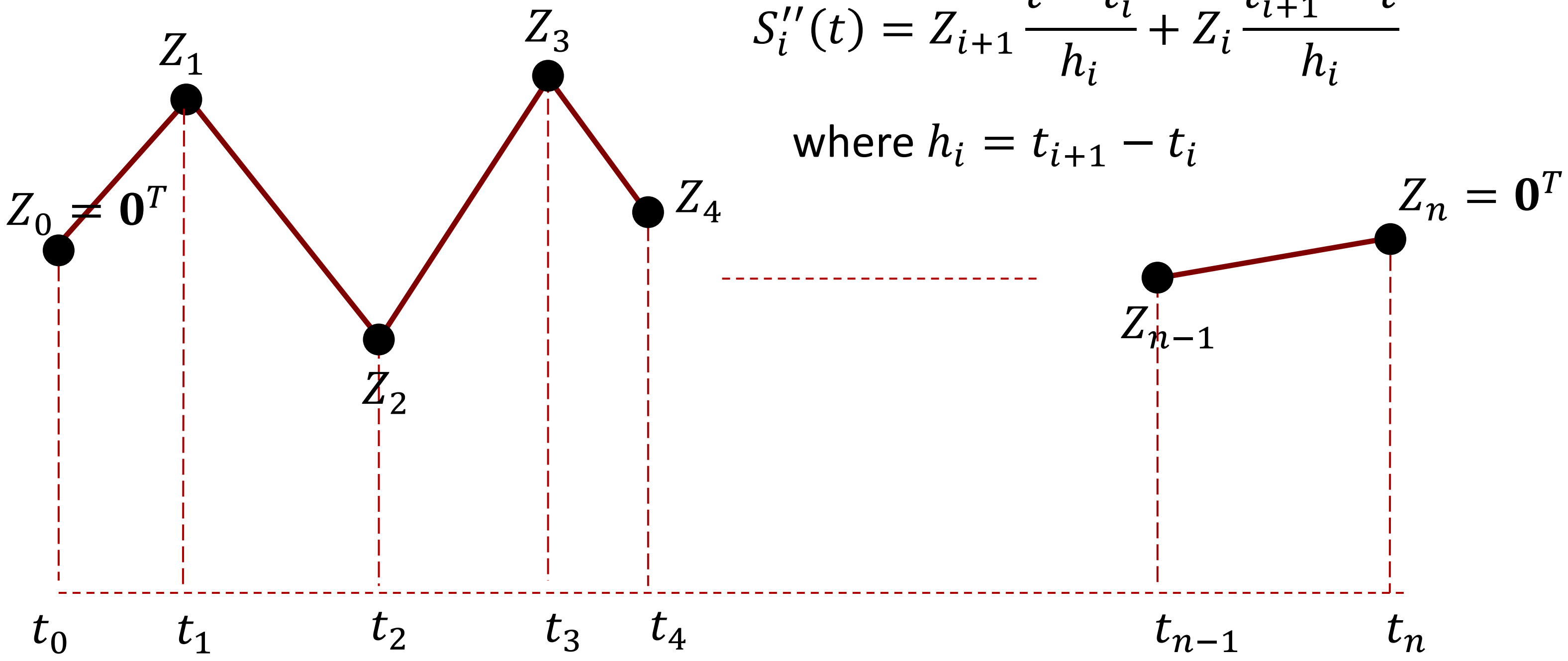
$$S_i''(t) = Z_i \frac{t - t_{i+1}}{t_i - t_{i+1}} + Z_{i+1} \frac{t - t_i}{t_{i+1} - t_i}$$



This derivation is largely inspired from Arne Morten Kvarving's slides on cubic splines

Computing Polynomials

- $S''(t)$ will be piecewise-linear



$$S_i''(t) = Z_{i+1}\frac{t - t_i}{h_i} + Z_i\frac{t_{i+1} - t}{h_i}$$

where $h_i = t_{i+1} - t_i$

Computing Polynomials

- At this point, we need to integrate twice to obtain $S(t)$

$$S_i''(t) = Z_{i+1}\frac{t - t_i}{h_i} + Z_i\frac{t_{i+1} - t}{h_i}$$

$$S_i'(t) = Z_{i+1}\frac{(t - t_i)^2}{2h_i} + Z_i\frac{(t_{i+1} - t)^2}{-2h_i} + C_i$$

$$S_i(t) = Z_{i+1}\frac{(t - t_i)^3}{6h_i} + Z_i\frac{(t_{i+1} - t)^3}{6h_i} + C_i t + D_i$$

$$S_i(t) = Z_{i+1}\frac{(t - t_i)^3}{6h_i} + Z_i\frac{(t_{i+1} - t)^3}{6h_i} + E_i(t - t_i) + F_i(t_{i+1} - t)$$

where $C_i = E_i - F_i$ and $D_i = F_i t_{i+1} - E_i t_i$

Computing Polynomials

- At this point, the only unknowns are Z_i, Z_{i+1}, E_i and F_i

$$S_i(t) = Z_{i+1}\frac{(t-t_i)^3}{6h_i} + Z_i\frac{(t_{i+1}-t)^3}{6h_i} + E_i(t-t_i) + F_i(t_{i+1}-t)$$

- Plug-in the values at the control points:

$$S_i(t_i) = \ P_i = Z_i\frac{{h_i}^2}{6} + F_i h_i$$

$$S_i(t_{i+1}) = \ P_{i+1} = Z_{i+1}\frac{{h_i}^2}{6} + E_i h_i$$

- From here, we can determine E_i and F_i

Computing Polynomials

- This gives us:

$$S_i(t) = \boldsymbol{Z_{i+1}} \frac{(t - t_i)^3}{6h_i} + \boldsymbol{Z_i} \frac{(t_{i+1} - t)^3}{6h_i} + \left(\frac{P_{i+1}}{h_i} - \frac{Z_{i+1}h_i}{6}\right)(t - t_i) + \left(\frac{P_i}{h_i} - \frac{Z_i h_i}{6}\right)(t_{i+1} - t)$$

- Finally, we need to compute the $\boldsymbol{Z_i}$ terms
- We know that $\boldsymbol{Z_0} = \boldsymbol{Z_n} = [0\ 0\ 0]^T$

Computing Polynomials

- We did not use the constraint that the first derivatives at the control points are equal; so take the derivative

$$\boldsymbol{S}_{\boldsymbol{i}}'(\boldsymbol{t}) = \boldsymbol{Z}_{\boldsymbol{i+1}} \frac{(t-t_i)^2}{2h_i} - \boldsymbol{Z}_{\boldsymbol{i}} \frac{(t_{i+1}-t)^2}{2h_i} + \underbrace{\frac{1}{h_i}(\boldsymbol{P}_{\boldsymbol{i+1}} - \boldsymbol{P}_{\boldsymbol{i}})}_{\boldsymbol{B}_{\boldsymbol{i}}} - \frac{h_i}{6}(\boldsymbol{Z}_{\boldsymbol{i+1}} - \boldsymbol{Z}_{\boldsymbol{i}})$$

$$\boldsymbol{S}_{\boldsymbol{i}}'(\boldsymbol{t}_{\boldsymbol{i}}) = -\boldsymbol{Z}_{\boldsymbol{i}} \frac{h_i}{2} + \boldsymbol{B}_{\boldsymbol{i}} - \frac{h_i}{6} \boldsymbol{Z}_{\boldsymbol{i+1}} + \frac{h_i}{6} \boldsymbol{Z}_{\boldsymbol{i}}$$

Computing Polynomials

- Repeat this for the previous (or the next) segment:

$$\boldsymbol{S}'_{\boldsymbol{i-1}}(\boldsymbol{t}) = \boldsymbol{Z_i}\frac{(t-t_{i-1})^2}{2h_{i-1}} - \boldsymbol{Z_{i-1}}\frac{(t_i-t)^2}{2h_{i-1}} + \underbrace{\frac{1}{h_{i-1}}(\boldsymbol{P_i}-\boldsymbol{P_{i-1}})}_{\boldsymbol{B_{i-1}}} - \frac{h_{i-1}}{6}(\boldsymbol{Z_i}-\boldsymbol{Z_{i-1}})$$

$$\boldsymbol{S}'_{\boldsymbol{i-1}}(\boldsymbol{t_i}) = \boldsymbol{Z_i}\frac{h_{i-1}}{2} + \boldsymbol{B_{i-1}} - \frac{h_{i-1}}{6}\boldsymbol{Z_i} + \frac{h_{i-1}}{6}\boldsymbol{Z_{i-1}}$$

Computing Polynomials

- Now equate the segments at the control points:

$$\boldsymbol{S}_{\boldsymbol{i}}'(\boldsymbol{t}_{\boldsymbol{i}}) = \boldsymbol{S}_{\boldsymbol{i-1}}'(\boldsymbol{t}_{\boldsymbol{i}})$$

$$-\boldsymbol{Z}_{\boldsymbol{i}}\frac{h_i}{2} + \boldsymbol{B}_{\boldsymbol{i}} - \frac{h_i}{6}\boldsymbol{Z}_{\boldsymbol{i+1}} + \frac{h_i}{6}\boldsymbol{Z}_{\boldsymbol{i}} = \boldsymbol{Z}_{\boldsymbol{i}}\frac{h_{i-1}}{2} + \boldsymbol{B}_{\boldsymbol{i-1}} - \frac{h_{i-1}}{6}\boldsymbol{Z}_{\boldsymbol{i}} + \frac{h_{i-1}}{6}\boldsymbol{Z}_{\boldsymbol{i-1}}$$

$$-3\boldsymbol{Z}_{\boldsymbol{i}}h_i + 6\boldsymbol{B}_{\boldsymbol{i}} - h_i\boldsymbol{Z}_{\boldsymbol{i+1}} + h_i\boldsymbol{Z}_{\boldsymbol{i}} = 3\boldsymbol{Z}_{\boldsymbol{i}}h_{i-1} + 6\boldsymbol{B}_{\boldsymbol{i-1}} - h_{i-1}\boldsymbol{Z}_{\boldsymbol{i}} + h_{i-1}\boldsymbol{Z}_{\boldsymbol{i-1}}$$

$$6(\boldsymbol{B}_{\boldsymbol{i}} - \boldsymbol{B}_{\boldsymbol{i-1}}) = h_{i-1}\boldsymbol{Z}_{\boldsymbol{i-1}} + 2(h_{i-1} + h_i)\boldsymbol{Z}_{\boldsymbol{i}} + h_i\boldsymbol{Z}_{\boldsymbol{i+1}}$$

Computing Polynomials

- We can set up *n-1* equations in this form and we also have *n-1* unknowns (from Z_1 to Z_{n-1})

$$6(\boldsymbol{B_i} - \boldsymbol{B_{i-1}}) = h_{i-1}\boldsymbol{Z_{i-1}} + 2(h_{i-1} + h_i)\boldsymbol{Z_i} + h_i\boldsymbol{Z_{i+1}}$$

- In the above equation, plug $i = 1 \dots n-1$, and solve the resulting system

Computing Polynomials

- Setup the system such that we have the $Ax = b$ form

$$\begin{bmatrix} v_1 & h_1 & 0 & \cdots & & \\ h_1 & v_2 & h_2 & & & \\ 0 & h_2 & v_3 & h_3 & & \\ \vdots & & & \ddots & & \\ & & & & v_{n-2} & h_{n-2} \\ & & & & h_{n-2} & v_{n-1} \end{bmatrix} \begin{bmatrix} Z_{1,x} \\ Z_{2,x} \\ Z_{3,x} \\ \vdots \\ \\ Z_{n-1,x} \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ \\ b_{n-1} \end{bmatrix}$$

$$h_i = t_{i+1} - t_i$$

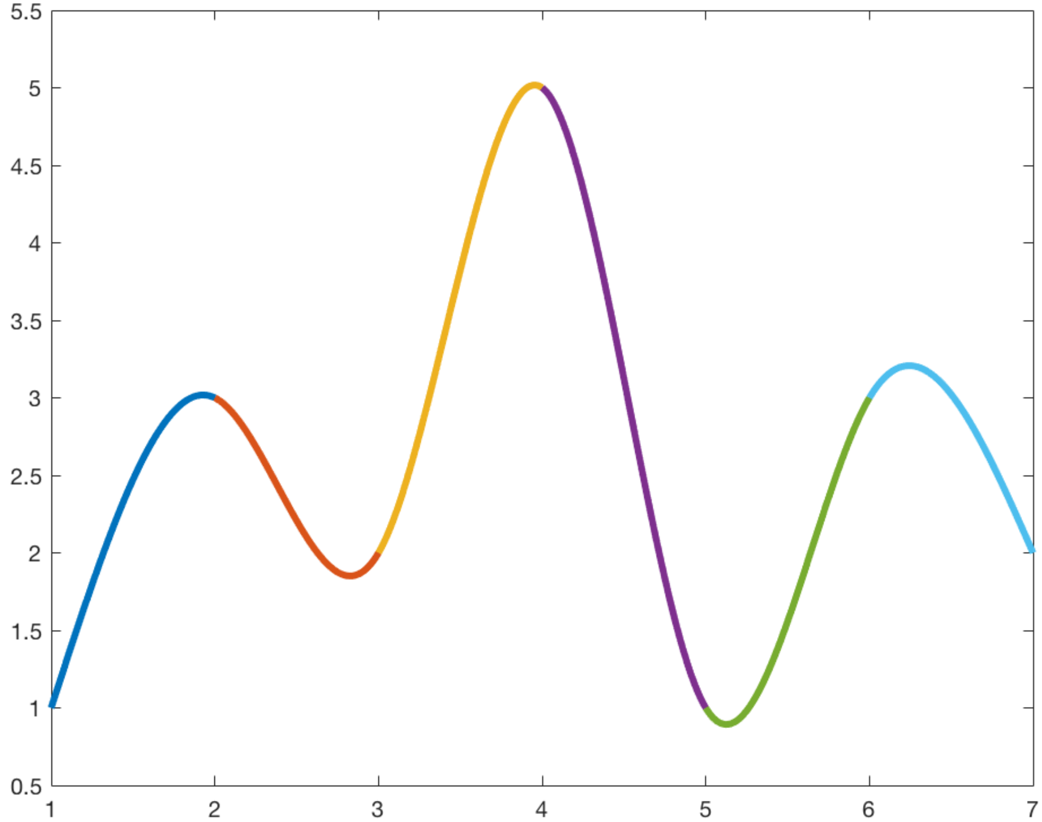
$$v_i = 2(h_{i-1} + h_i)$$

$$b_i = 6(B_{i,x} - B_{i-1,x})$$

$$B_{i,x} = \frac{1}{h_i}\left(P_{i+1,x} - P_{i,x}\right)$$

- Note that we are solving for the x-components
- We need to solve for y- and z-components if our curve is 3 dimensional

Sample Output



P = {(1, 1), (2, 3), (3, 2), (4, 5), (5, 1), (6, 3), (7, 2)}
T = {0, 1, 2, 3, 4, 5, 6}

Parametric Bicubic Surfaces

- Generalization of parametric cubic curves
- Recall $Q(t) = TMG$
- First replace t by s such that $Q(s) = SMG$
- Now allow the points in G to vary along a curve parametrized by t

$$Q(s,t) = SMG(t)$$

$$G_{1x}(t) \quad G_{1y}(t) \quad G_{1z}(t)$$

$$Q(s,t) = SM \begin{bmatrix} G_1(t) \\ G_2(t) \\ G_3(t) \\ G_4(t) \end{bmatrix}$$

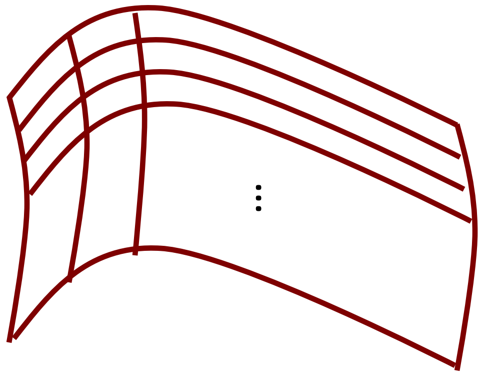
Parametric Bicubic Surfaces

- We can setup separate equations for *x*, *y*, and *z*:

$$Q_x(s,t) = SM\begin{bmatrix} G_{1x}(t) \\ G_{2x}(t) \\ G_{3x}(t) \\ G_{4x}(t) \end{bmatrix} \qquad Q_y(s,t) = SM\begin{bmatrix} G_{1y}(t) \\ G_{2y}(t) \\ G_{3y}(t) \\ G_{4y}(t) \end{bmatrix} \qquad Q_z(s,t) = SM\begin{bmatrix} G_{1z}(t) \\ G_{2z}(t) \\ G_{3z}(t) \\ G_{4z}(t) \end{bmatrix}$$

Parametric Bicubic Surfaces

- Now for a fixed $t = t_1$, $Q(s, t_1)$ is a curve because $G(t_1)$ is constant
- Allowing t to take on a different value, t_2, where $t_2 - t_1$ is very small, $Q(s, t_2)$ is a slightly different curve
- Repeating this arbitrarily many times gives you a large set of curves, which is our surface



Called bicubic if $G_i(t)$ are themselves cubic

Derivation

- Assume that $G_i(t)$ themselves are defined by:

$$G_i(t) = TMZ_i$$

$$Z_{i1x} \quad Z_{i1y} \quad Z_{i1z}$$

$$G_i(t) = TM \begin{bmatrix} Z_{i1} \\ Z_{i2} \\ Z_{i3} \\ Z_{i4} \end{bmatrix}$$

$$G_{ix}(t) = TM \begin{bmatrix} Z_{i1x} \\ Z_{i2x} \\ Z_{i3x} \\ Z_{i4x} \end{bmatrix} \qquad G_{iy}(t) = TM \begin{bmatrix} Z_{i1y} \\ Z_{i2y} \\ Z_{i3y} \\ Z_{i4y} \end{bmatrix} \qquad G_{iz}(t) = TM \begin{bmatrix} Z_{i1z} \\ Z_{i2z} \\ Z_{i3z} \\ Z_{i4z} \end{bmatrix}$$

Derivation

- Remember that we have:

$$Q_x(s,t) = SM \begin{bmatrix} G_{1x}(t) \\ G_{2x}(t) \\ G_{3x}(t) \\ G_{4x}(t) \end{bmatrix} \qquad G_{ix}(t) = TM \begin{bmatrix} Z_{i1x} \\ Z_{i2x} \\ Z_{i3x} \\ Z_{i4x} \end{bmatrix}$$

- To combine them into a single equation, take the transpose of $G_{ix}(t)$, which is equal to itself due to its being a scalar:

$$G_{1x}(t)^T = G_{1x}(t) = [Z_{11x} \quad Z_{11x} \quad Z_{13x} \quad Z_{14x}]M^T T^T$$

$$G_{2x}(t)^T = G_{2x}(t) = [Z_{21x} \quad Z_{21x} \quad Z_{23x} \quad Z_{24x}]M^T T^T$$

$$G_{3x}(t)^T = G_{3x}(t) = [Z_{31x} \quad Z_{31x} \quad Z_{33x} \quad Z_{34x}]M^T T^T$$

$$G_{4x}(t)^T = G_{4x}(t) = [Z_{41x} \quad Z_{41x} \quad Z_{43x} \quad Z_{44x}]M^T T^T$$

Derivation

- Remember that we have:

$$Q_x(s,t) = SM \begin{bmatrix} G_{1x}(t) \\ G_{2x}(t) \\ G_{3x}(t) \\ G_{4x}(t) \end{bmatrix} \qquad G_{ix}(t) = TM \begin{bmatrix} Z_{i1x} \\ Z_{i2x} \\ Z_{i3x} \\ Z_{i4x} \end{bmatrix}$$

- To combine them into a single equation, take the transpose of $G_{ix}(t)$, which is equal to itself due to its being a scalar:

$$\begin{bmatrix} G_{1x}(t) \\ G_{2x}(t) \\ G_{3x}(t) \\ G_{4x}(t) \end{bmatrix} = \begin{bmatrix} Z_{11x} & Z_{11x} & Z_{13x} & Z_{14x} \\ Z_{21x} & Z_{21x} & Z_{23x} & Z_{24x} \\ Z_{31x} & Z_{31x} & Z_{33x} & Z_{34x} \\ Z_{41x} & Z_{41x} & Z_{43x} & Z_{44x} \end{bmatrix} M^T T^T$$

Derivation

- This gives us:

$$Q_x(s,t) = S \underbrace{\begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \\ m_{41} & m_{42} & m_{43} & m_{44} \end{bmatrix}}_{M} \underbrace{\begin{bmatrix} Z_{11x} & Z_{11x} & Z_{13x} & Z_{14x} \\ Z_{21x} & Z_{21x} & Z_{23x} & Z_{24x} \\ Z_{31x} & Z_{31x} & Z_{33x} & Z_{34x} \\ Z_{41x} & Z_{41x} & Z_{43x} & Z_{44x} \end{bmatrix}}_{G_x} M^T T^T$$

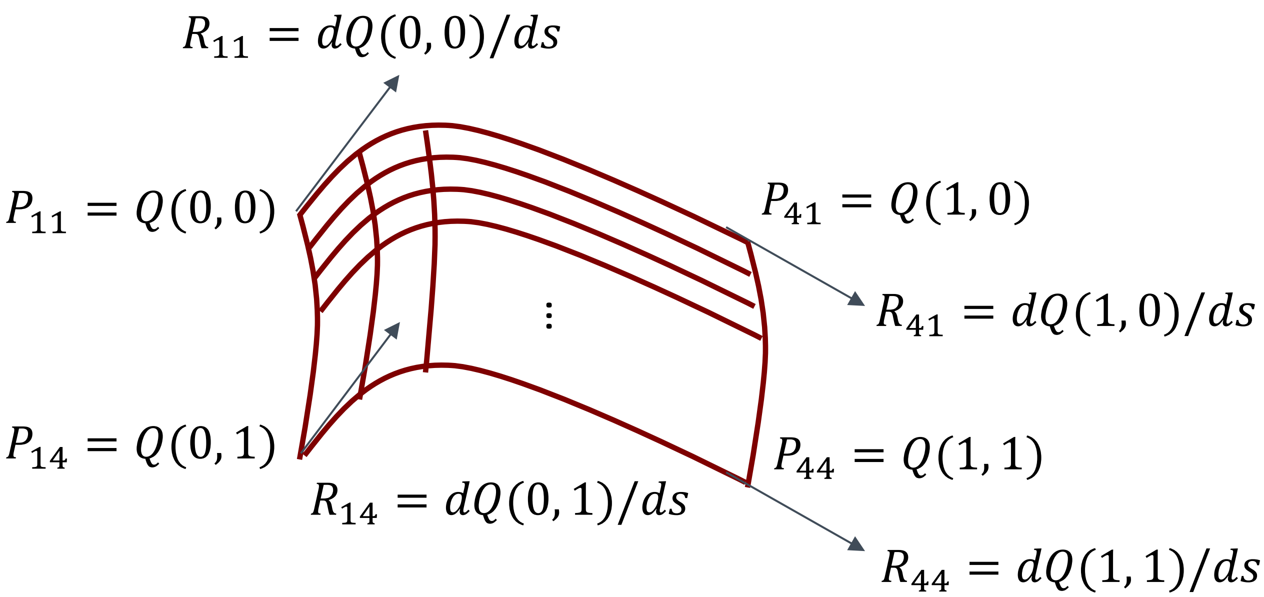
- Similarly for *y* and *z*:

$$Q_y(s,t) = SMG_yM^TT^T$$

$$Q_z(s,t) = SMG_zM^TT^T$$

Hermite Surfaces

- For Hermite surfaces, M is the Hermite basis matrix
- The elements of the geometry matrix (G_x, G_y, G_z) store how each component changes with respect to t:



Hermite Surfaces

- How the starting point (P_1) changes with respect to t:

$$G_x = \begin{bmatrix} Q_x(0,0) & Q_x(0,1) & \frac{dQ_x(0,0)}{dt} & \frac{dQ_x(0,1)}{dt} \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \end{bmatrix}$$

Hermite Surfaces

- How the end point (P_4) changes with respect to t:

$$G_x = \begin{bmatrix} \dots & \dots & \dots & \dots \\ Q_x(1,0) & Q_x(1,1) & \dfrac{dQ_x(1,0)}{dt} & \dfrac{dQ_x(1,1)}{dt} \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \end{bmatrix}$$

Hermite Surfaces

- How the starting tangent vector (R_1), defined with respect to s, changes with respect to t:

$$G_x = \begin{bmatrix} \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ \frac{dQ_x(0,0)}{ds} & \frac{dQ_x(0,1)}{ds} & \frac{d^2Q_x(0,0)}{dsdt} & \frac{d^2Q_x(0,1)}{dsdt} \\ \cdots & \cdots & \cdots & \cdots \end{bmatrix}$$

Hermite Surfaces

- How the ending tangent vector (R_4), defined with respect to s, changes with respect to t:

$$G_x = \begin{bmatrix} \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ \frac{dQ_x(1,0)}{ds} & \frac{dQ_x(1,1)}{ds} & \frac{d^2Q_x(1,0)}{dsdt} & \frac{d^2Q_x(1,1)}{dsdt} \end{bmatrix}$$

Hermite Surfaces

- So the entire geometry matrix looks like:

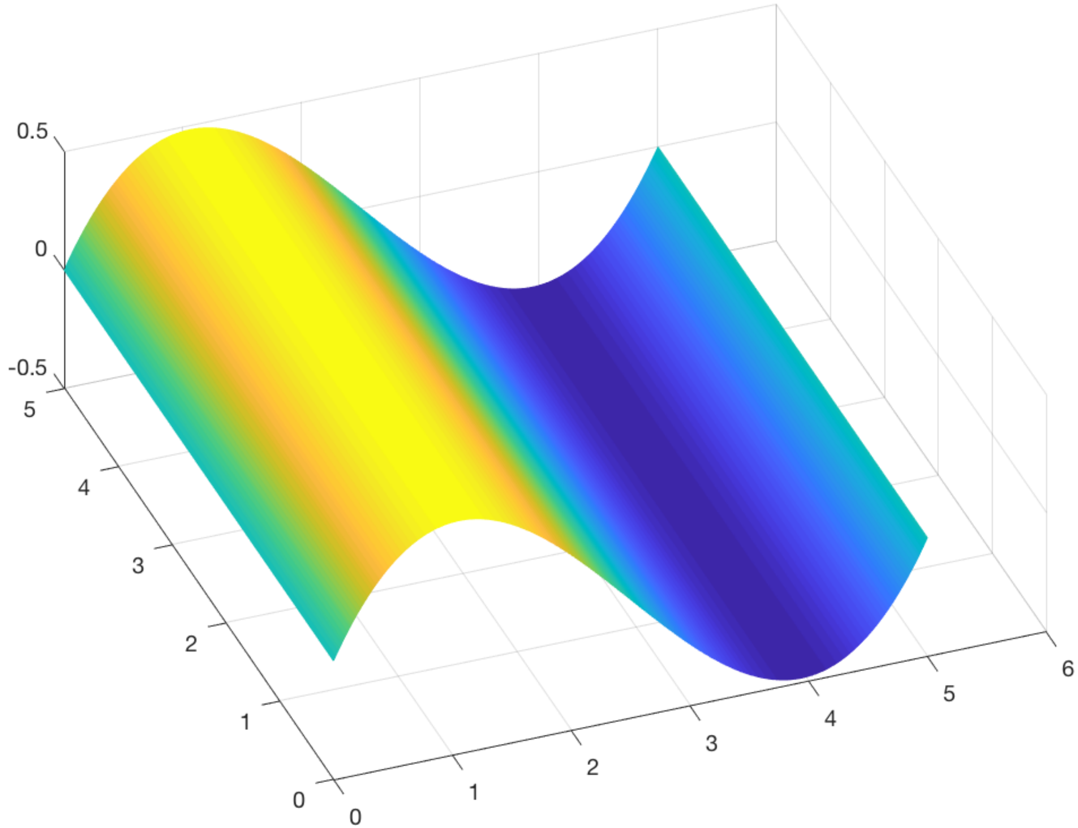
$$G_x = \begin{bmatrix} Q_x(0,0) & Q_x(0,1) & \frac{dQ_x(0,0)}{dt} & \frac{dQ_x(0,1)}{dt} \\ Q_x(1,0) & Q_x(1,1) & \frac{dQ_x(1,0)}{dt} & \frac{dQ_x(1,1)}{dt} \\ \frac{dQ_x(0,0)}{ds} & \frac{dQ_x(0,1)}{ds} & \frac{d^2Q_x(0,0)}{dsdt} & \frac{d^2Q_x(0,1)}{dsdt} \\ \frac{dQ_x(1,0)}{ds} & \frac{dQ_x(1,1)}{ds} & \frac{d^2Q_x(1,1)}{dsdt} & \frac{d^2Q_x(1,1)}{dsdt} \end{bmatrix}$$

Hermite Surfaces

$$G_x = \begin{bmatrix} 0 & 5 & 5 & 5 \\ 0 & 5 & 5 & 5 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$G_y = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 5 & 5 & 0 & 0 \\ 5 & 5 & 0 & 0 \\ 5 & 5 & 0 & 0 \end{bmatrix}$$

$$G_z = \begin{bmatrix} 0 & 0 & 5 & 5 \\ 0 & 0 & 5 & 5 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$



Bezier Surfaces

- Remember that Bezier curves was defined using Bernstein polynomials:

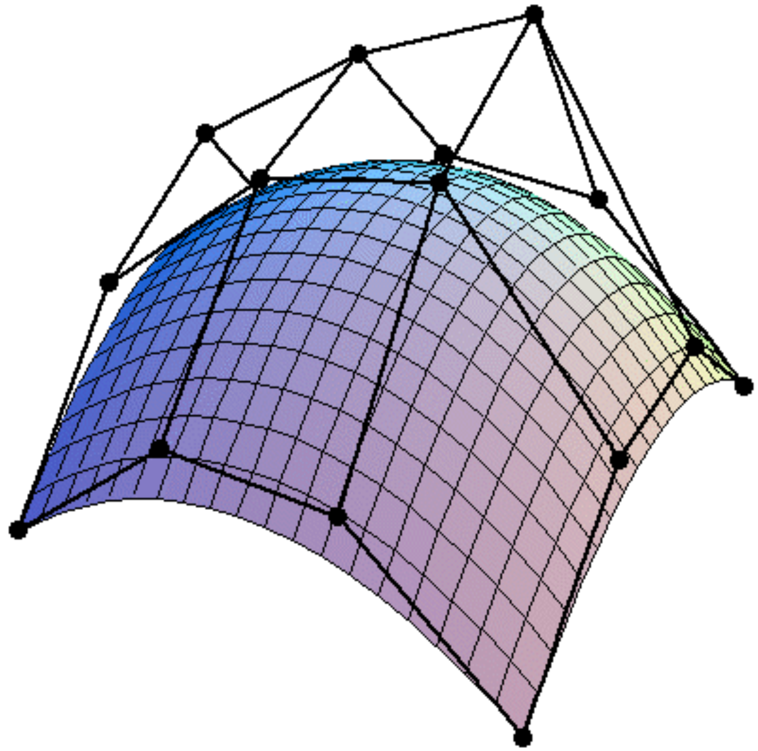
$$Q(t) = \sum_{i=0}^{n} B_{i,n}(t) P_{i+1} \qquad B_{i,n}(t) = \binom{n}{i} t^i (1-t)^{n-i}$$

- Their extension to surfaces is straightforward:

$$Q(s,t) = \sum_{i=0}^{n} \sum_{j=0}^{m} B_{i,n}(s) B_{i,m}(t) P_{i+1}$$

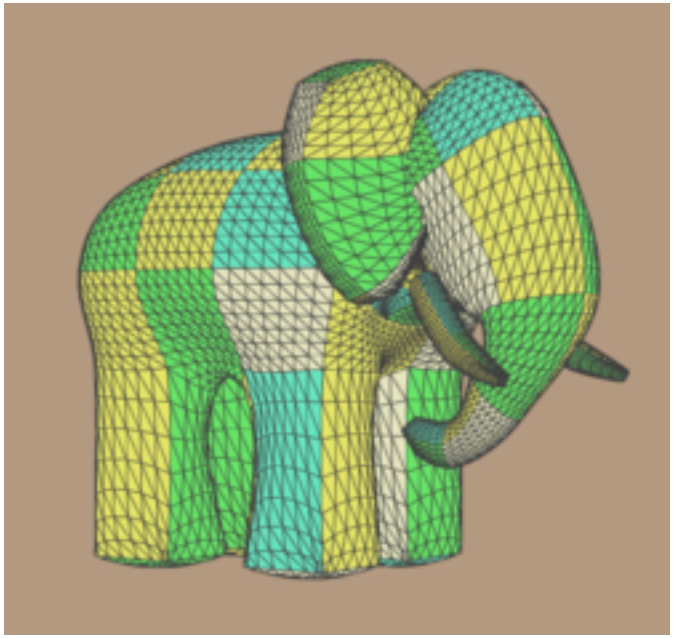
Bezier Surfaces

- Similar to a Bezier curve, a Bezier surface interpolates the end points and approximates the interior control points



Bezier Surfaces

- Complex models can be created using Bezier surfaces
- In such models, the entire surface is composed of multiple Bezier surfaces, known as patches



Gumbo Model

Bezier Surfaces

- Such patches allows tessellating a surface at the desired level of detail depending on viewing distance or other parameters
- OpenGL tessellation shaders provide hardware support for this