



Curves, Splines, Patches, Basis

CS535

Daniel G. Aliaga
Department of Computer Science
Purdue University

[some slides based on Bruce Cohen and David Sklar, and Jack van Wijk]

Planar (Triangular) Patches vs Curved Patches



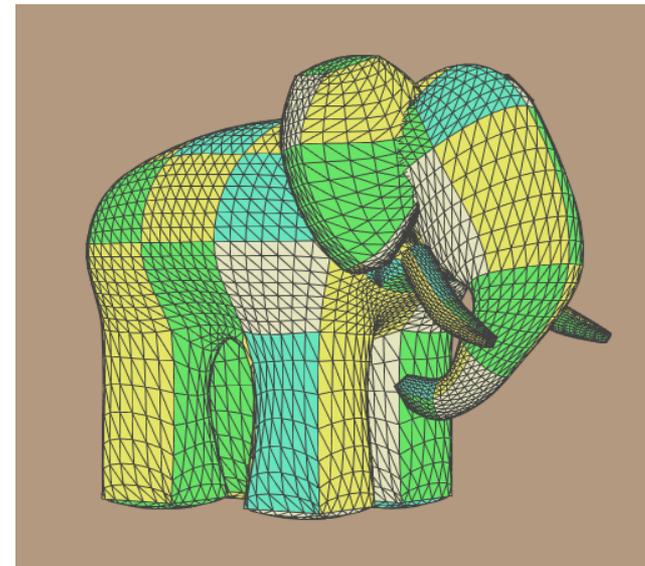
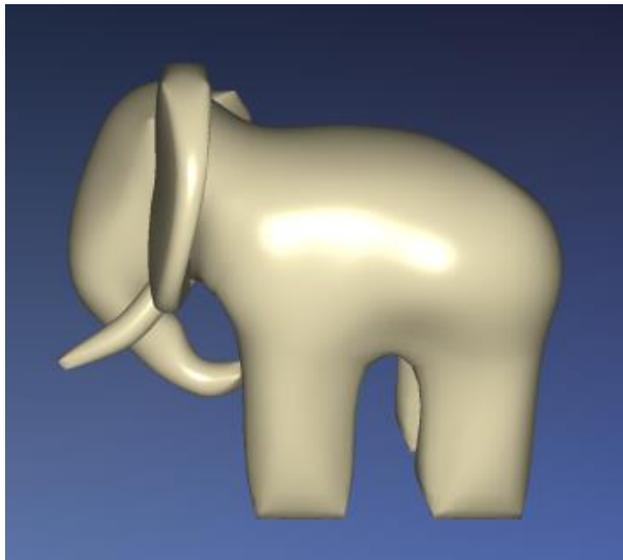
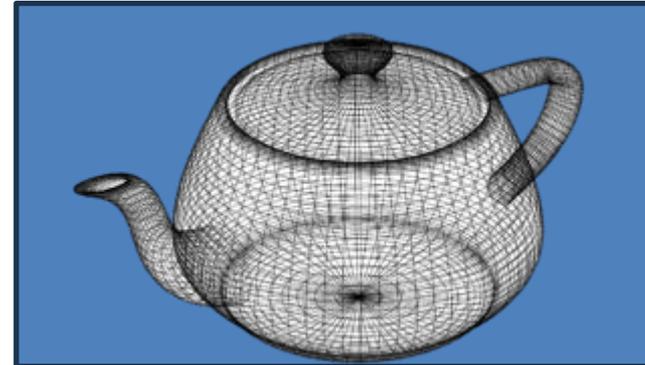


Ice Sculptures and more

- Carlo Sequin @ Berkeley et al.
 - https://people.eecs.berkeley.edu/~sequin/PAPERS/Bridges04_Snowball.pdf



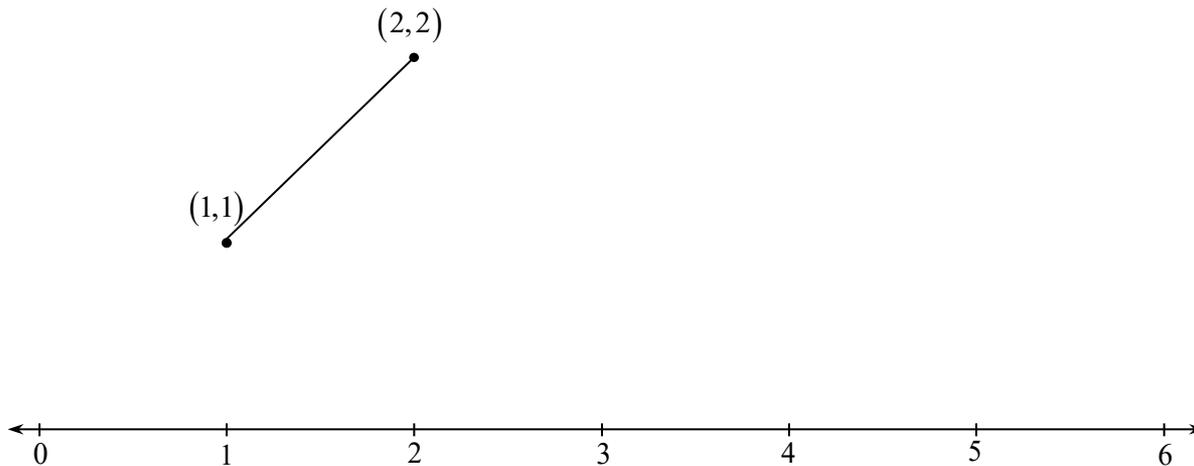
Planar (Triangular) Patches vs Curved Patches



Connecting the Dots



What is the formula for this line?

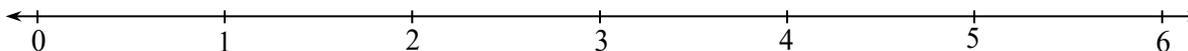
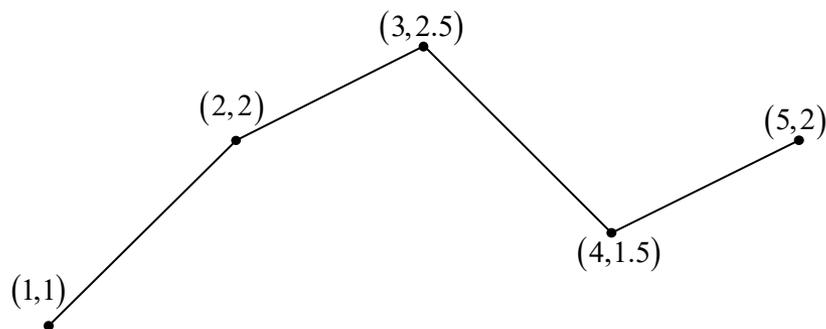


$$f(x) = x; f(x) = (2 - x) * 1 + (x - 1) * 2$$

Connecting the Dots



Write a formula for a piecewise linear function that interpolates five given data points

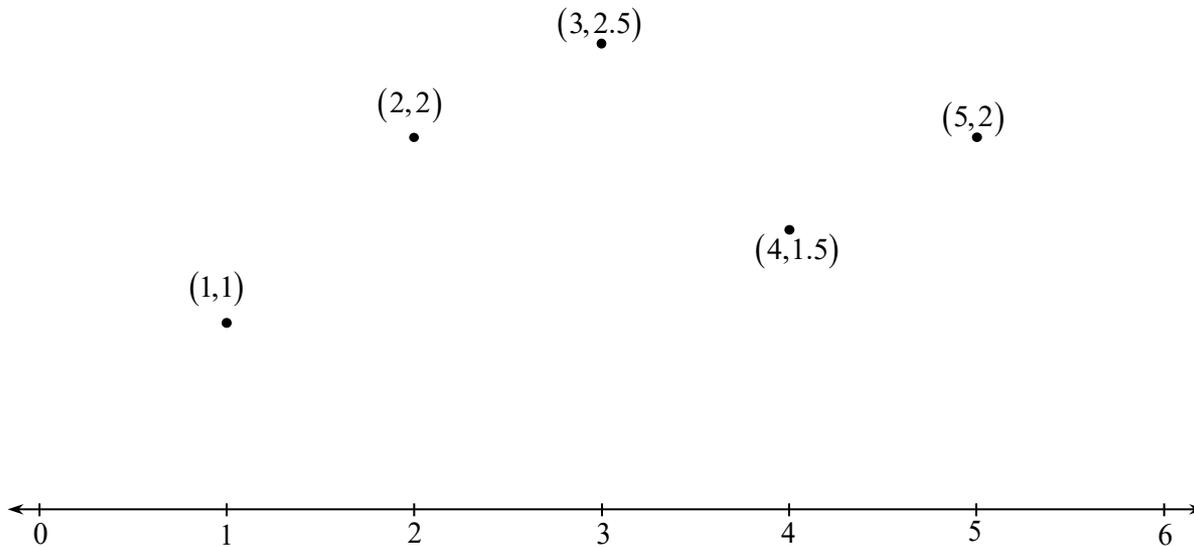


$$f(x) = \begin{cases} x & \text{if } 1 \leq x \leq 2 \\ \frac{1}{2}x + 1 & \text{if } 2 \leq x \leq 3 \\ -x + \frac{11}{2} & \text{if } 3 \leq x \leq 4 \\ \frac{1}{2}x - \frac{1}{2} & \text{if } 4 \leq x \leq 5 \end{cases}$$

Connecting the Dots



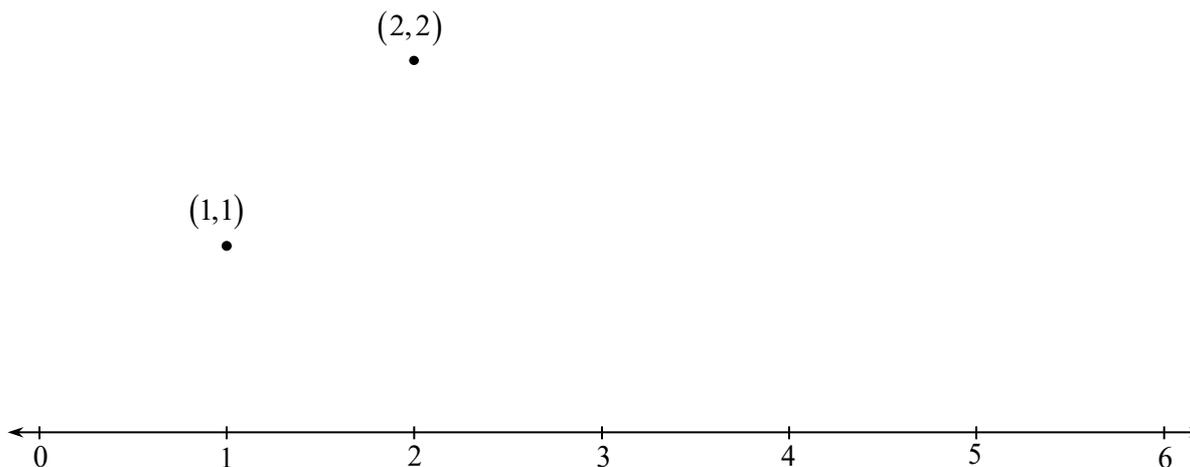
Alternatives?



Connecting the Dots



Interpolating Polynomial



$$f(1) = 1 \text{ and } f(2) = 2$$

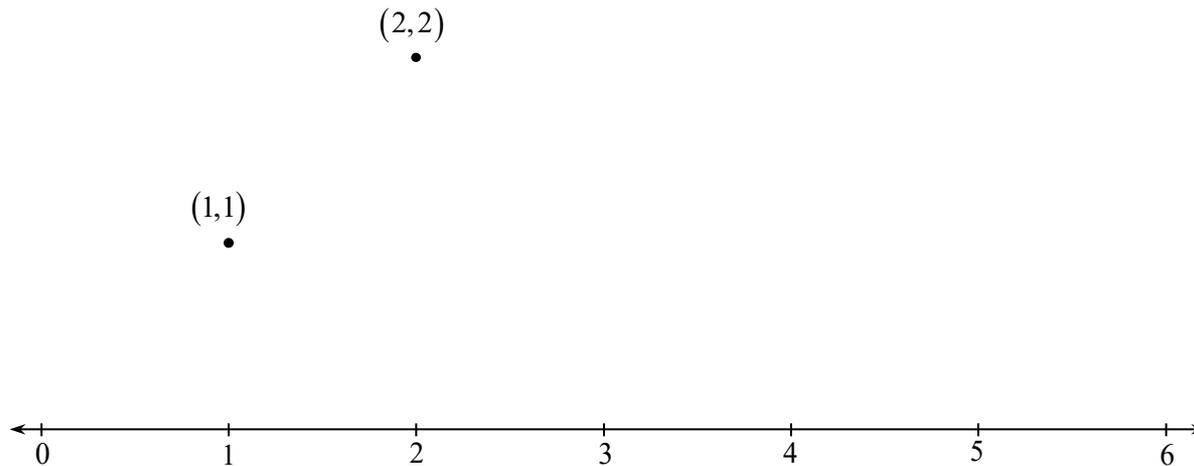
$$h_0(x) = \frac{x-x_1}{x_0-x_1} \text{ is } 1 \text{ when } x = 1 \text{ and is } 0 \text{ when } x = 2$$

$$h_1(x) = \frac{x-x_0}{x_1-x_0} \text{ is } 1 \text{ when } x = 2 \text{ and is } 0 \text{ when } x = 1$$

Connecting the Dots



Interpolating Polynomial



$$f(1) = 1 \text{ and } f(2) = 2$$

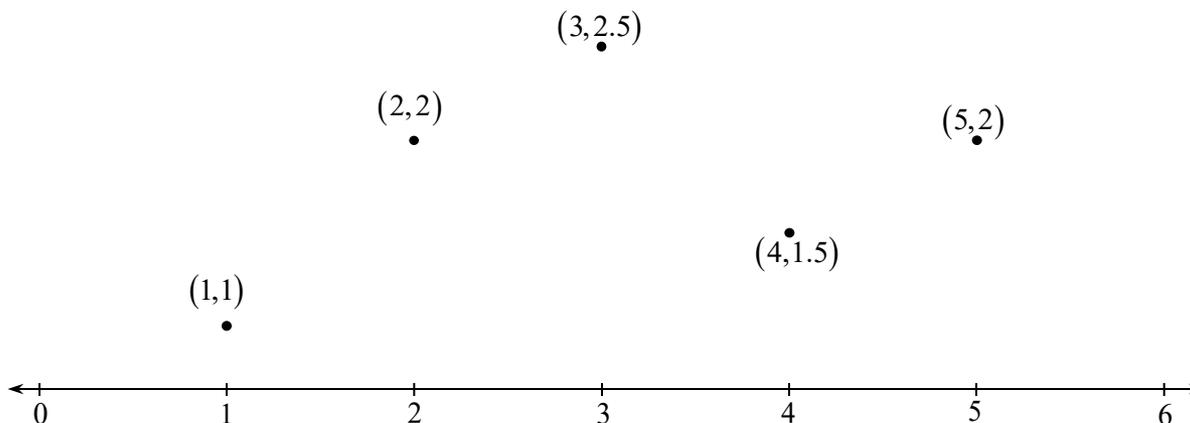
$$f(x) = y_0 h_0(x) + y_1 h_1(x)$$

$$\text{thus } f(x_0) = y_0 \text{ and } f(x_1) = y_1$$

Connecting the Dots



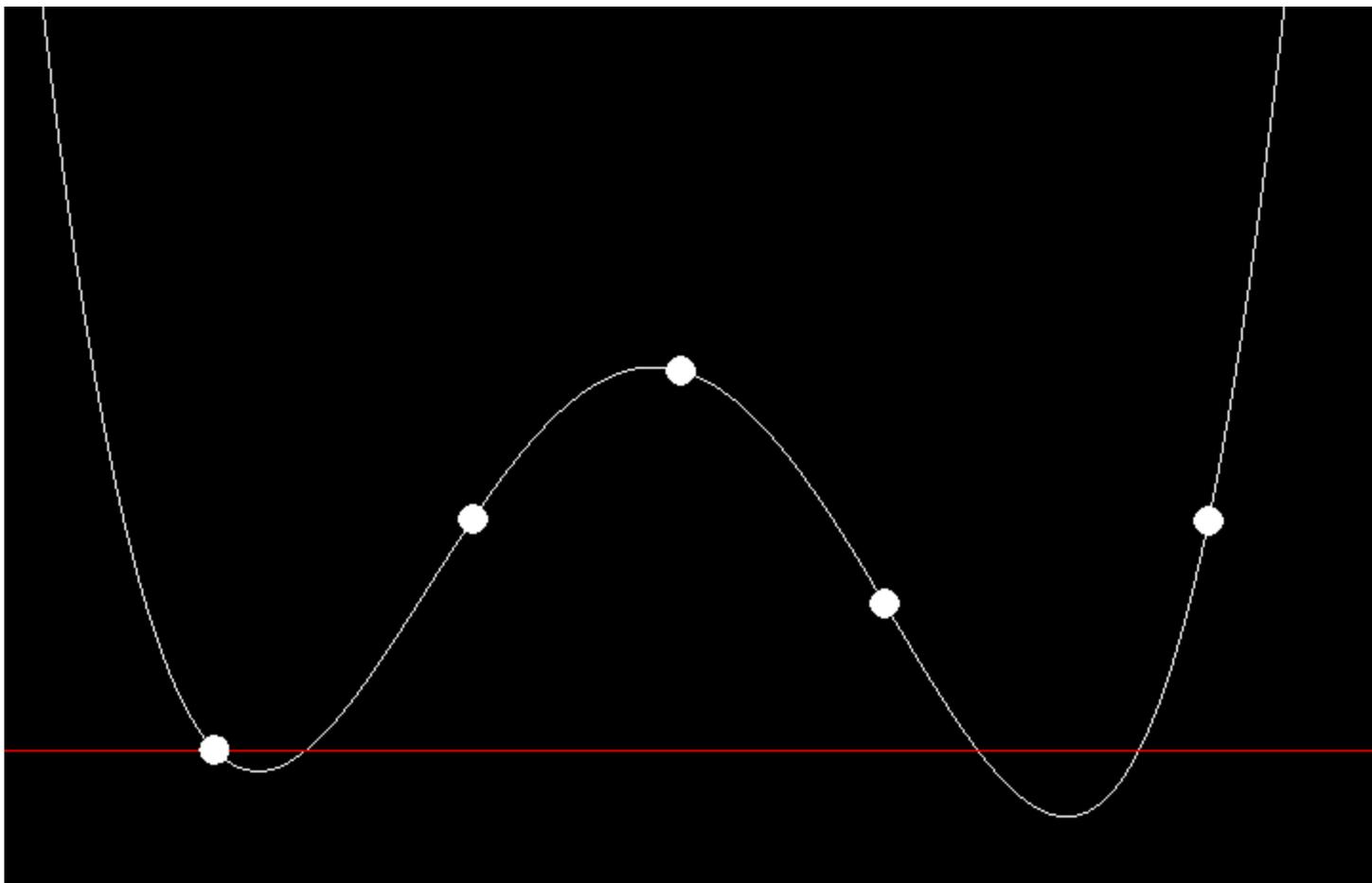
Interpolating Polynomial



$$\phi_i(x) = \frac{\prod_{j \neq i} (x - x_j)}{\prod_{j \neq i} (x_i - x_j)} \quad \text{Lagrange Interpolation Basis}$$

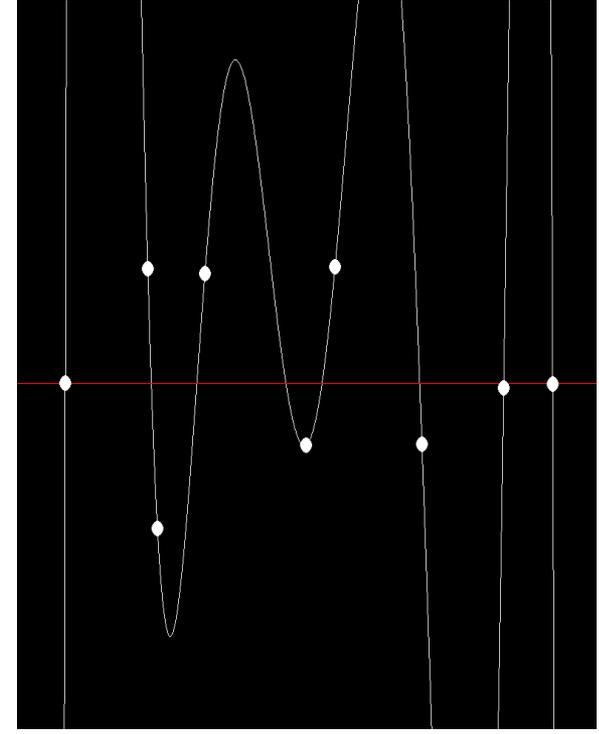
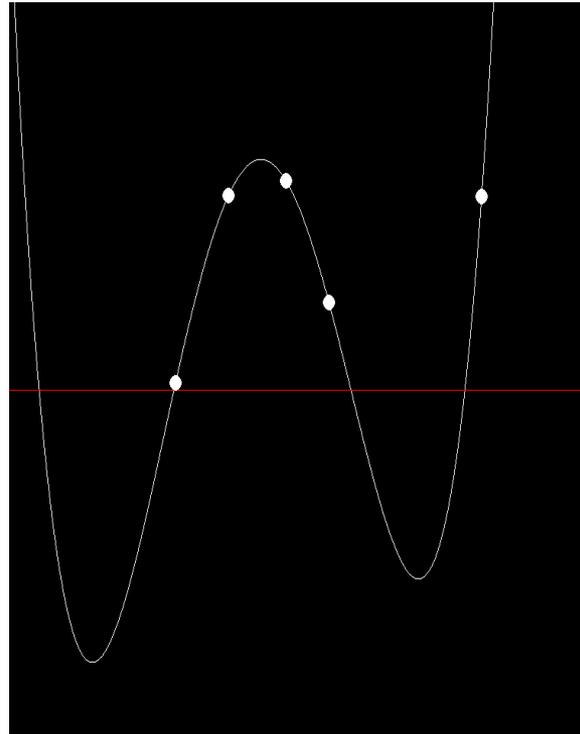
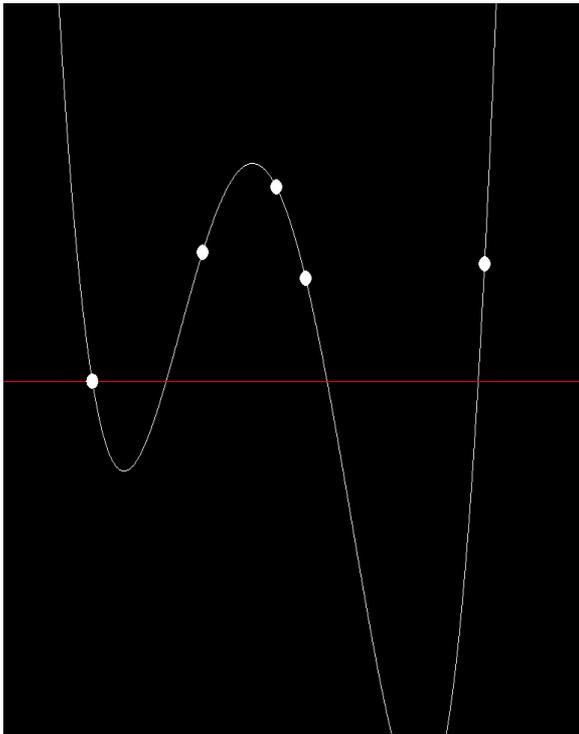
$$\phi_i(x_l) = 1 \text{ when } l = i, \text{ else } 0$$

$$f(x) = \sum_i y_i \phi_i(x) \quad \text{Lagrange Polynomial}$$



<https://l-e-webb.github.io/interpolate/webdist/index.html>

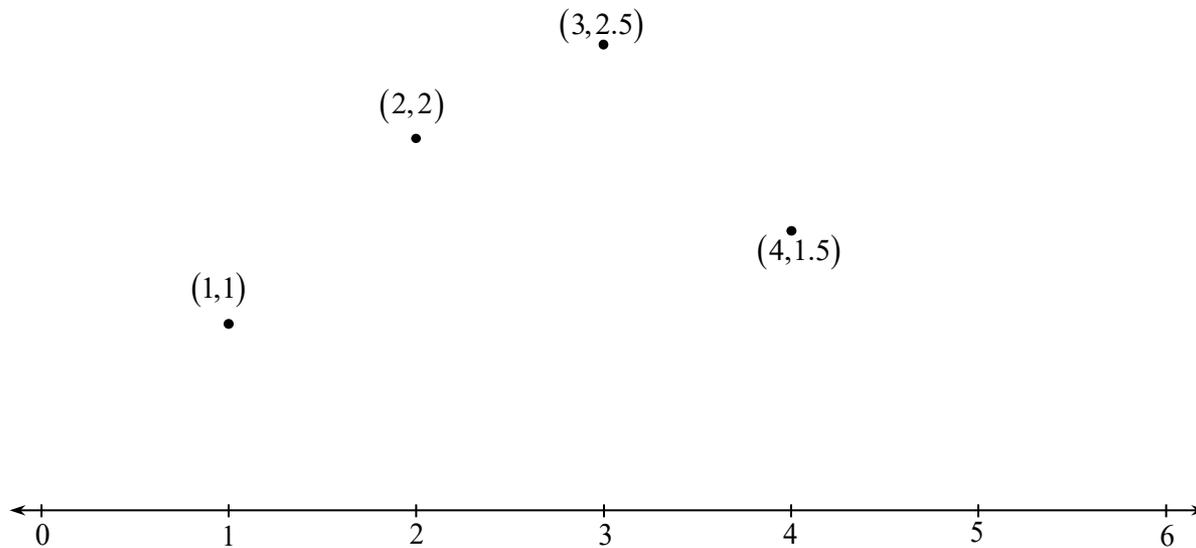
But not very stable...



Connecting the Dots



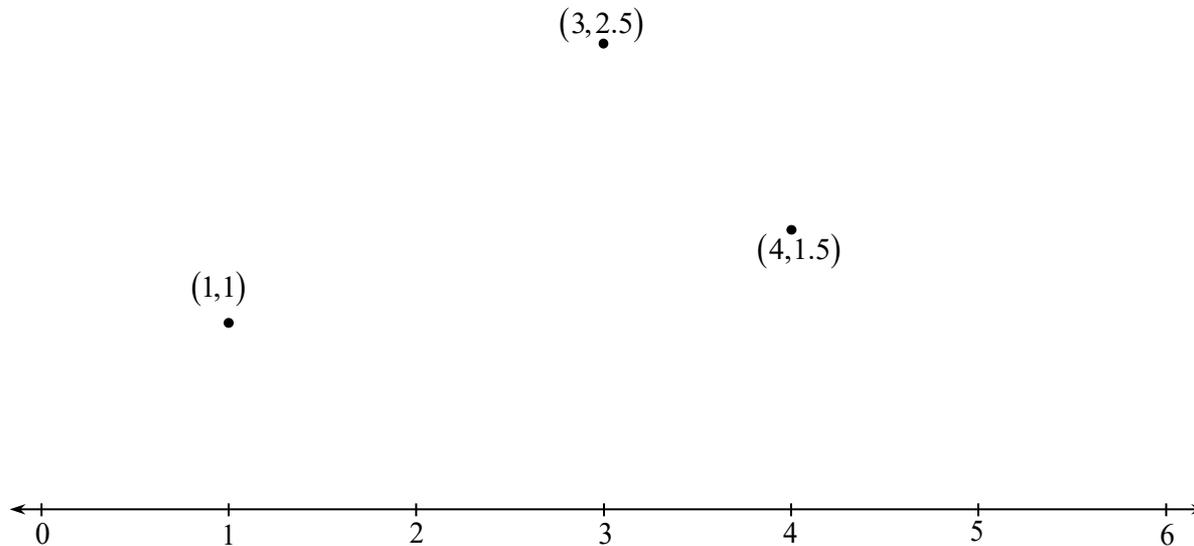
Alternatives?



Connecting the Dots



Bezier Curves



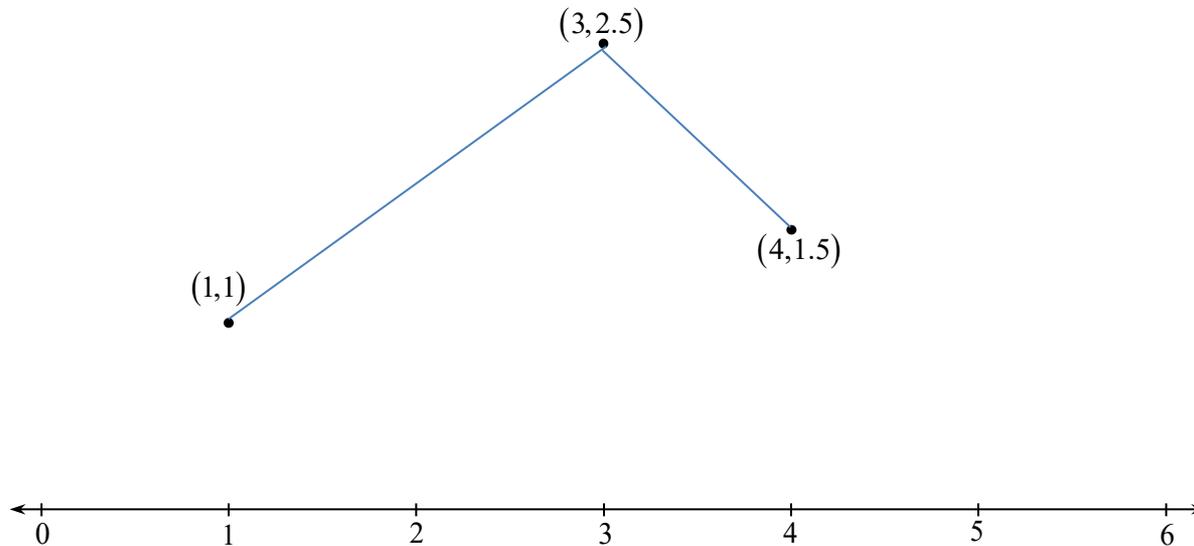
History:

- Mathematical basis established in 1912 using Bernstein polynomials
- Paul De Casteljaou used for Citroen's in 1959 (patented, released in 1980s)
- Pierre Bezier used for Renault in 1960s (widely publicized)

Connecting the Dots



Bezier Curves



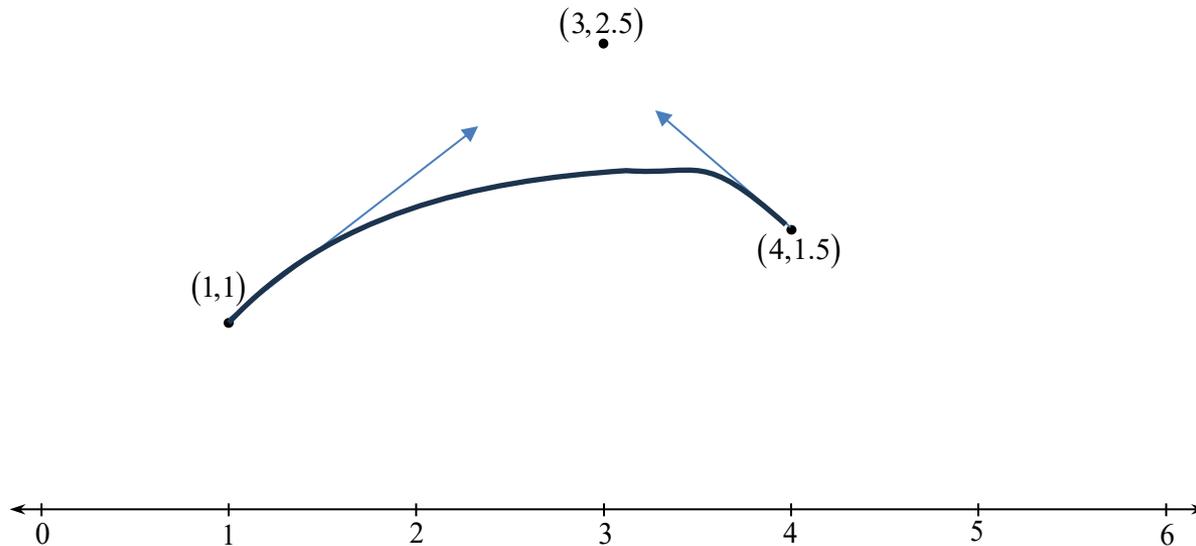
History:

- Mathematical basis established in 1912 using Bernstein polynomials
- Paul De Casteljaou used for Citroen's in 1959 (patented, released in 1980s)
- Pierre Bezier used for Renault in 1960s (widely publicized)

Connecting the Dots



Bezier Curves



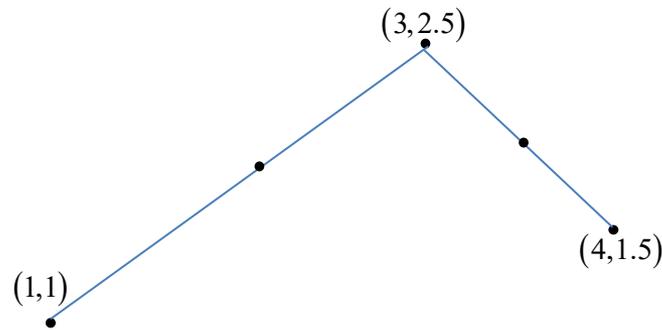
History:

- Mathematical basis established in 1912 using Bernstein polynomials
- Paul De Casteljau used for Citroen's in 1959 (patented, released in 1980s)
- Pierre Bezier used for Renault in 1960s (widely publicized)

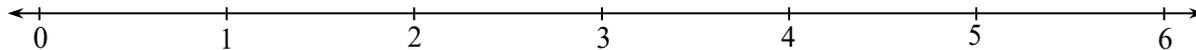
Connecting the Dots



Bezier Curves



If two (end) control points, then “first mid-point” is the new (mid) control point...



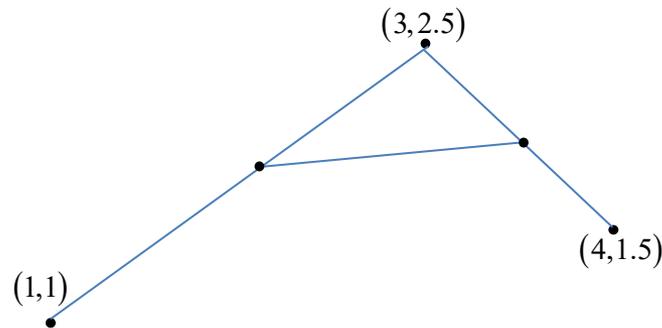
History:

- Mathematical basis established in 1912 using Bernstein polynomials
- Paul De Casteljaou used for Citroen’s in 1959 (patented, released in 1980s)
- Pierre Bezier used for Renault in 1960s (widely publicized)

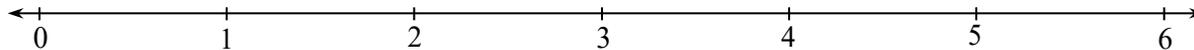
Connecting the Dots



Bezier Curves



If two (end) control points, then
“first mid-point” is the new (mid)
control point...



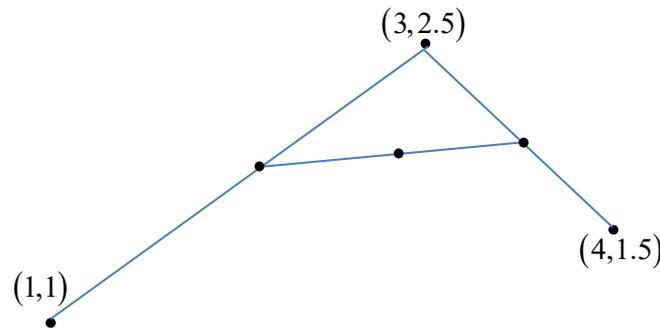
History:

- Mathematical basis established in 1912 using Bernstein polynomials
- Paul De Casteljaou used for Citroen's in 1959 (patented, released in 1980s)
- Pierre Bezier used for Renault in 1960s (widely publicized)

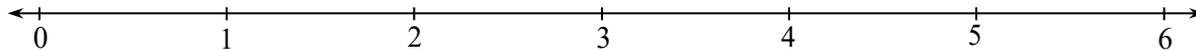
Connecting the Dots



Bezier Curves



If two (end) control points, then “first mid-point” is the new (mid) control point...



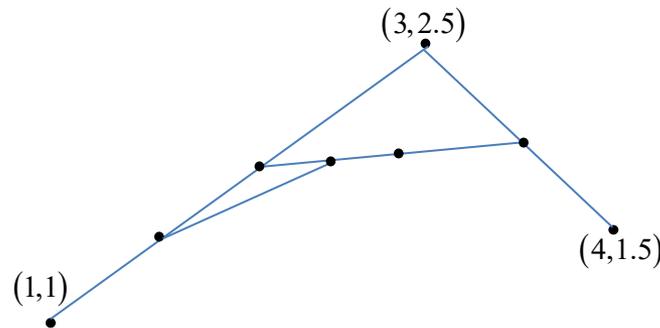
History:

- Mathematical basis established in 1912 using Bernstein polynomials
- Paul De Casteljaou used for Citroen’s in 1959 (patented, released in 1980s)
- Pierre Bezier used for Renault in 1960s (widely publicized)

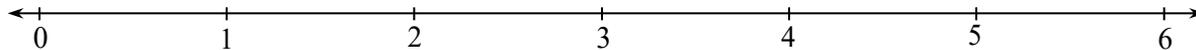
Connecting the Dots



Bezier Curves



If two (end) control points, then “first mid-point” is the new (mid) control point...



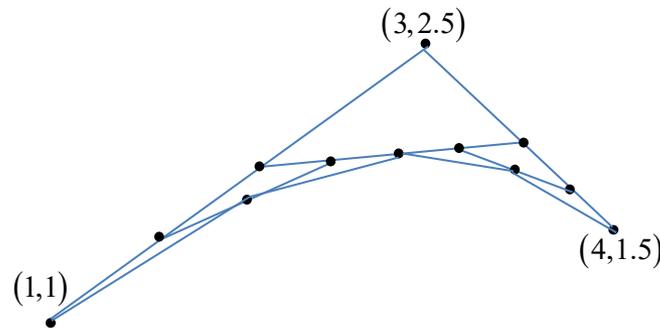
History:

- Mathematical basis established in 1912 using Bernstein polynomials
- Paul De Casteljaou used for Citroen’s in 1959 (patented, released in 1980s)
- Pierre Bezier used for Renault in 1960s (widely publicized)

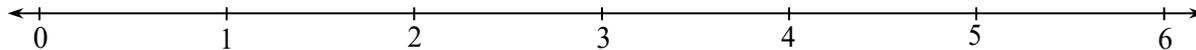
Connecting the Dots



Bezier Curves



If two (end) control points, then "first mid-point" is the new (mid) control point...



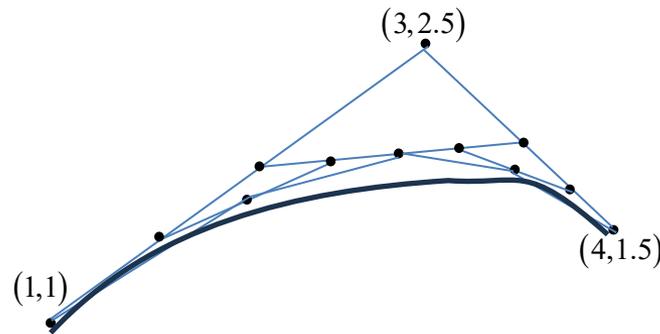
History:

- Mathematical basis established in 1912 using Bernstein polynomials
- Paul De Casteljaou used for Citroen's in 1959 (patented, released in 1980s)
- Pierre Bezier used for Renault in 1960s (widely publicized)

Connecting the Dots



Bezier Curves



If two (end) control points, then “first mid-point” is the new (mid) control point...

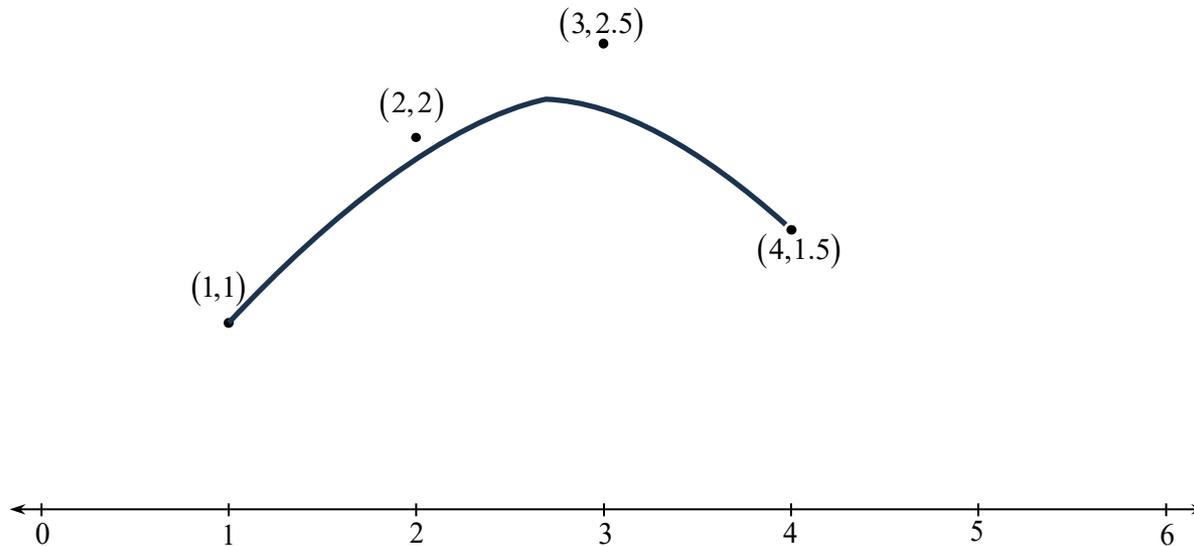
History:

- Mathematical basis established in 1912 using Bernstein polynomials
- Paul De Casteljau used for Citroen’s in 1959 (patented, released in 1980s)
- Pierre Bezier used for Renault in 1960s (widely publicized)

Connecting the Dots



Bezier Curves



Types ($[0 \leq t \leq 1]$):

Linear: $B(t) = (1 - t)p_0 + tp_1$

Quadratic: $B(t) = p_1 + (1 - t)^2(p_0 - p_1) + t^2(p_2 - p_1)$

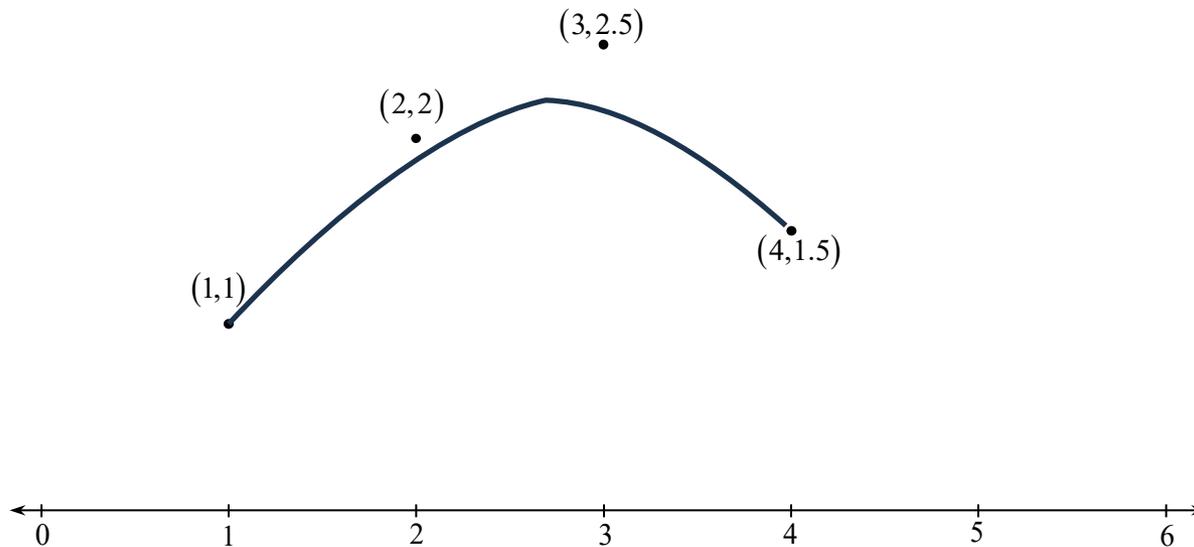
Cubic: $B(t) = (1 - t)B_{p_0,p_1,p_2} + tB_{p_1,p_2,p_3}(t)$

$$B(t) = (1 - t)^3 p_0 + 3(1 - t)^2 t p_1 + 3(1 - t) t^2 p_2 + t^3 p_3$$

Connecting the Dots



Bezier Curves



Cubic: $B(t) = (1-t)^3 p_0 + 3(1-t)^2 t p_1 + 3(1-t)t^2 p_2 + t^3 p_3$

Bernstein polynomials:

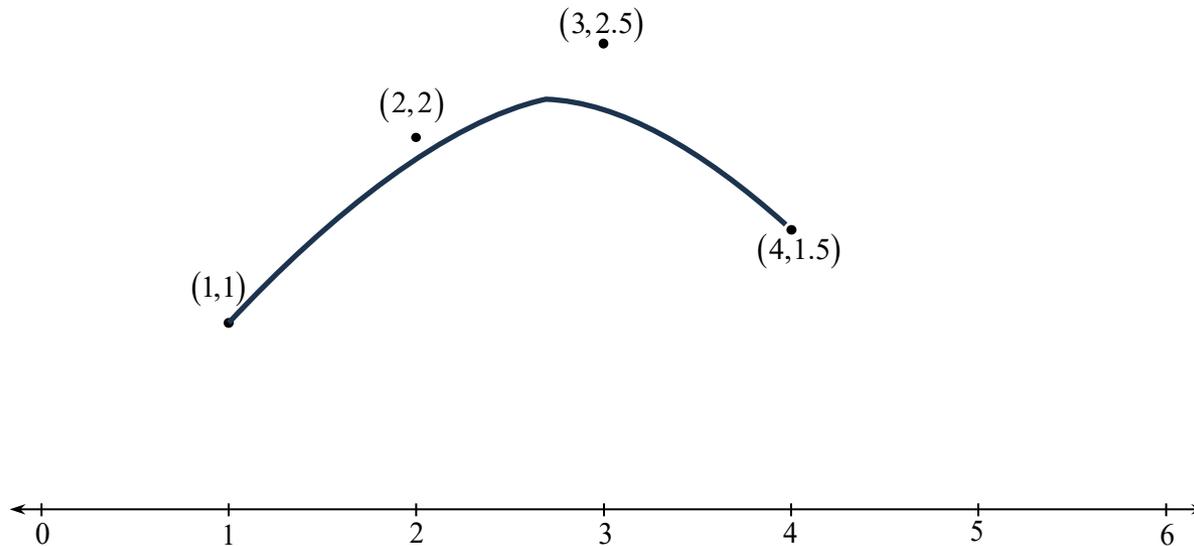
$$b_{\nu,n}(x) = \binom{n}{\nu} x^{\nu} (1-x)^{n-\nu}, \text{ for } \nu = 0, \dots, n,$$

where $\binom{n}{\nu}$ is a **binomial coefficient**.

Connecting the Dots



Bezier Curves



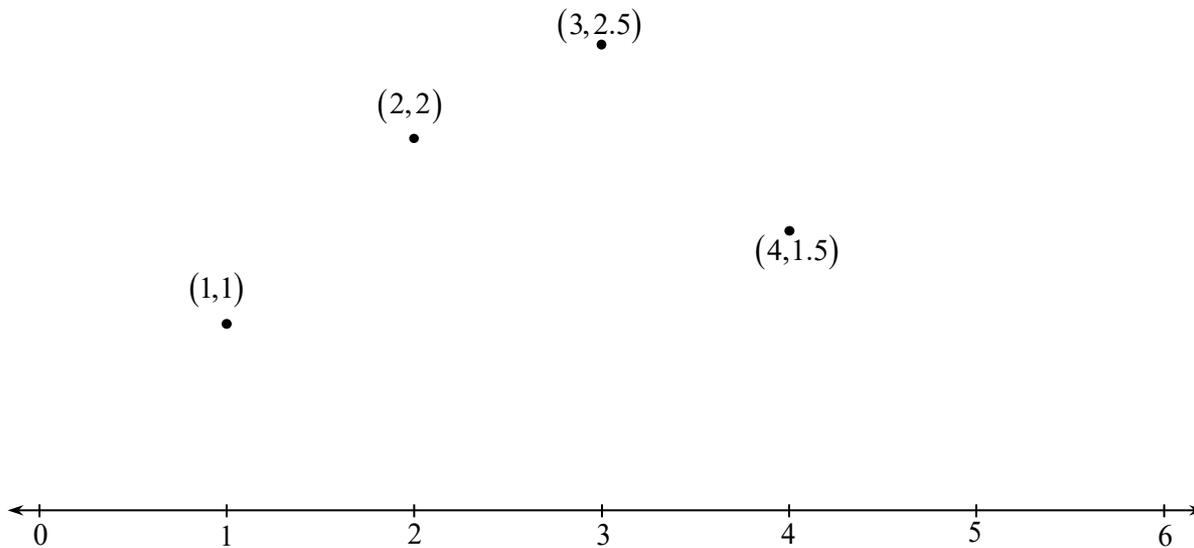
Properties:

- Curve begins at p_0 and ends at p_3 (and does not necessarily pass thru p_1 and p_2)
- Start and end is tangent to first/last section
- Curves have the variation diminishing property – do not undulate more than its control points

Connecting the Dots



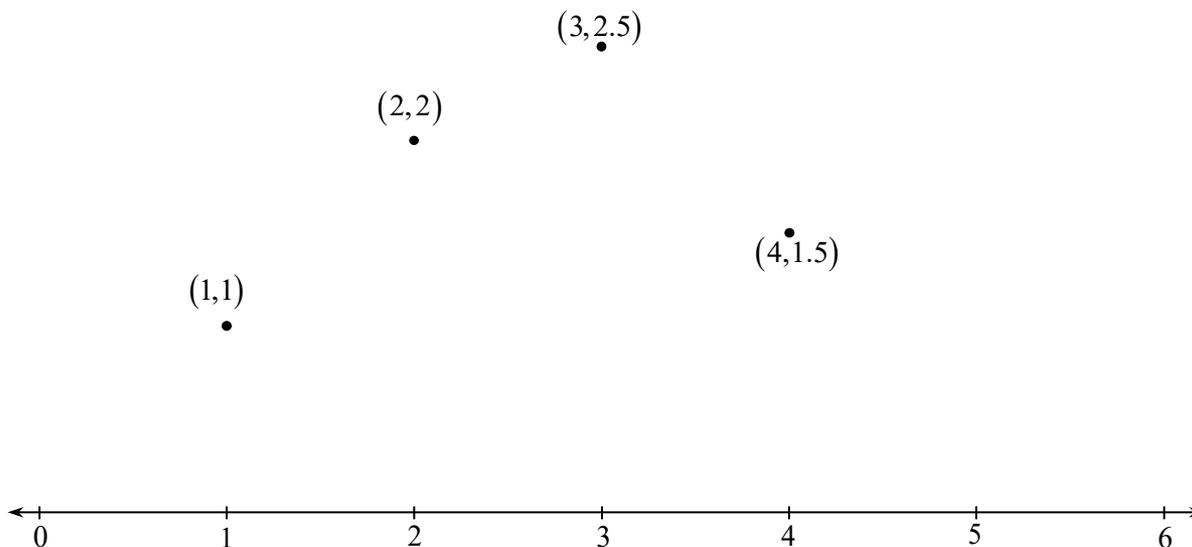
Alternatives?



Connecting the Dots



B-splines (Basis Splines)

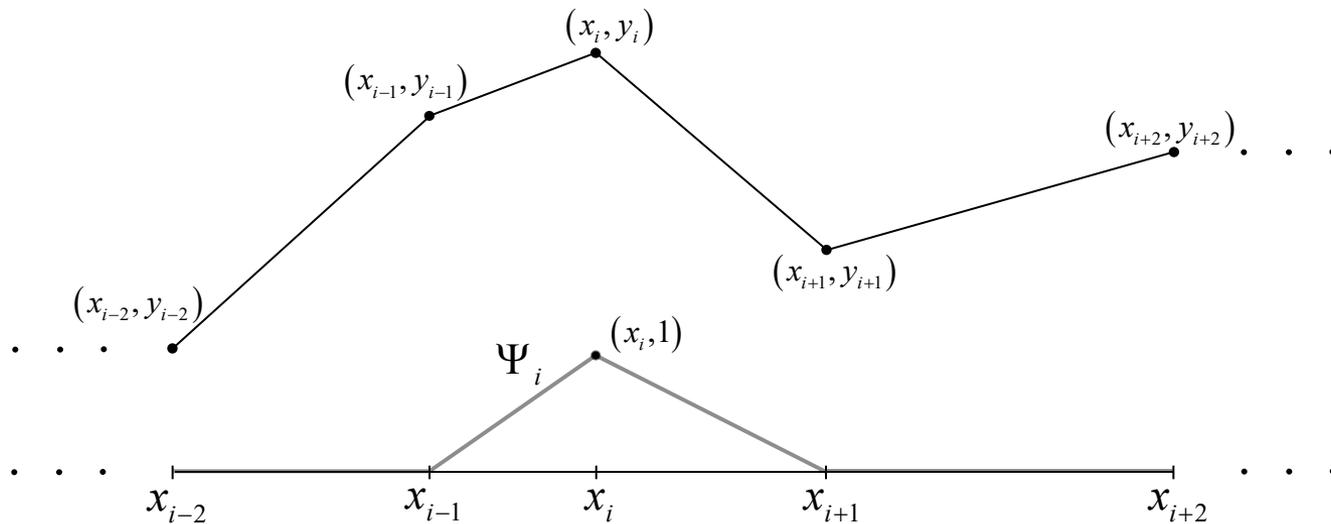


A spline uses a set of basis functions to express the line as a sum of weighted components (not necessarily Bernstein Polynomials):

$$f(x) = y_1\Psi_1(x) + y_2\Psi_2(x) + \cdots + y_n\Psi_n(x) = \sum_{i=1}^n y_i\Psi_i(x) \quad , \text{ for all } x_1 \leq x \leq x_n$$



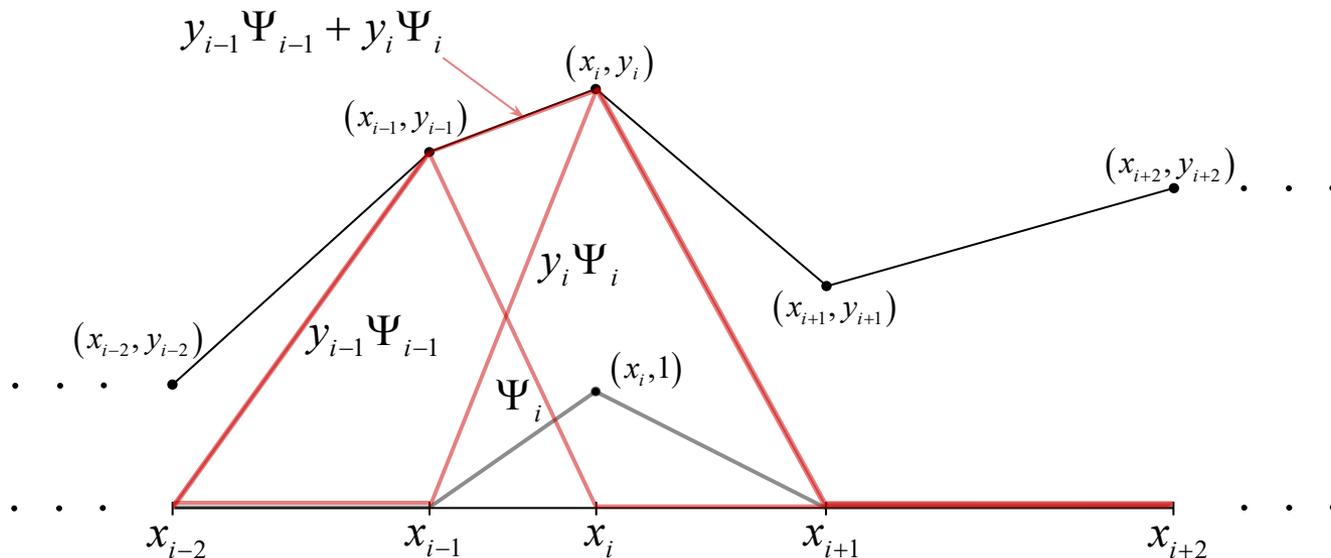
B-splines: With linear basis functions...





A closer look at a linear combination of basis functions

$$f(x) = y_1\Psi_1(x) + y_2\Psi_2(x) + \cdots + y_n\Psi_n(x) = \sum_{i=1}^n y_i\Psi_i(x)$$



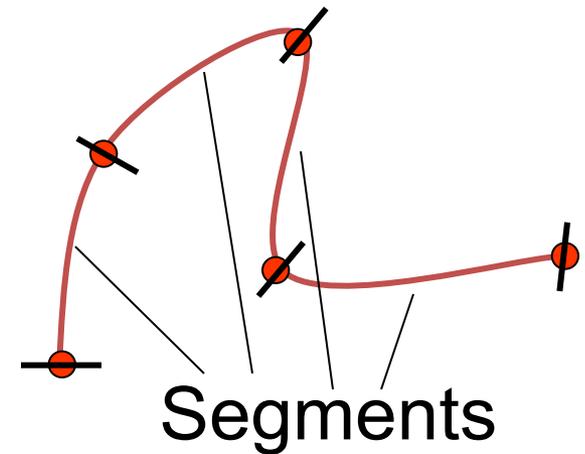


Cubic Splines

Sequence of segments defined by :

$$\mathbf{P}(u) = \begin{pmatrix} x(u) \\ y(u) \\ z(u) \end{pmatrix} = \begin{pmatrix} a_x u^3 + b_x u^2 + c_x u + d_x \\ a_y u^3 + b_y u^2 + c_y u + d_y \\ a_z u^3 + b_z u^2 + c_z u + d_z \end{pmatrix}$$

with $0 \leq u \leq 1$.



Cubic Splines (in 2D)



Let $p(x) = ax^3 + bx^2 + cx + d$ so $p'(x) = 3ax^2 + 2bx + c$

Using the data we have

$$\begin{aligned} p(x_1) &= ax_1^3 + bx_1^2 + cx_1 + d = y_1 \\ p(x_2) &= ax_2^3 + bx_2^2 + cx_2 + d = y_2 \\ p'(x_1) &= 3ax_1^2 + 2bx_1 + c = y_1' \\ p'(x_2) &= 3ax_2^2 + 2bx_2 + c = y_2' \end{aligned}$$

We have 4 equations in the four unknown coefficients.
So we can solve for a , b , c , and d .



Cubic Spline

Representation Options

$$\text{Var. 1: } \mathbf{P}(u) = \begin{pmatrix} u^3 & u^2 & u & 1 \end{pmatrix} \begin{pmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \\ d_x & d_y & d_z \end{pmatrix}$$

$$\text{Var. 2: } \mathbf{P}(u) = \begin{pmatrix} u^3 & u^2 & u & 1 \end{pmatrix} \begin{pmatrix} M_{00} & M_{01} & M_{02} & M_{03} \\ M_{10} & M_{11} & M_{12} & M_{13} \\ M_{20} & M_{21} & M_{22} & M_{23} \\ M_{30} & M_{31} & M_{32} & M_{33} \end{pmatrix} \begin{pmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \end{pmatrix}$$

$$\text{Var. 3: } \mathbf{P}(u) = \sum_{k=0}^3 B_k(u) \mathbf{P}_k$$



Cubic Splines

Variant 2 :

$$\mathbf{P}(u) = \begin{pmatrix} u^3 & u^2 & u & 1 \end{pmatrix} \begin{pmatrix} M_{00} & M_{01} & M_{02} & M_{03} \\ M_{10} & M_{11} & M_{12} & M_{13} \\ M_{20} & M_{21} & M_{22} & M_{23} \\ M_{30} & M_{31} & M_{32} & M_{33} \end{pmatrix} \begin{pmatrix} P_{0x} & P_{0y} & P_{0z} \\ P_{1x} & P_{1y} & P_{1z} \\ P_{2x} & P_{2y} & P_{2z} \\ P_{3x} & P_{3y} & P_{3z} \end{pmatrix}$$

$$= \mathbf{U} \mathbf{M}_{\text{spline}} \begin{pmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \end{pmatrix} = \mathbf{U} \mathbf{M}_{\text{spline}} \mathbf{M}_{\text{geom}}$$

↑
Control points or
Control vectors

Matrix $\mathbf{M}_{\text{spline}}$: 'translates' geometric info to coefficients



Bézier Spline Curves

Cubic Bézier spline :

$$\mathbf{P}(u) = (1-u)^3 \mathbf{P}_0 + 3u(1-u)^2 \mathbf{P}_1 + 3u^2(1-u) \mathbf{P}_2 + u^3 \mathbf{P}_3$$

In matrix representation :

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



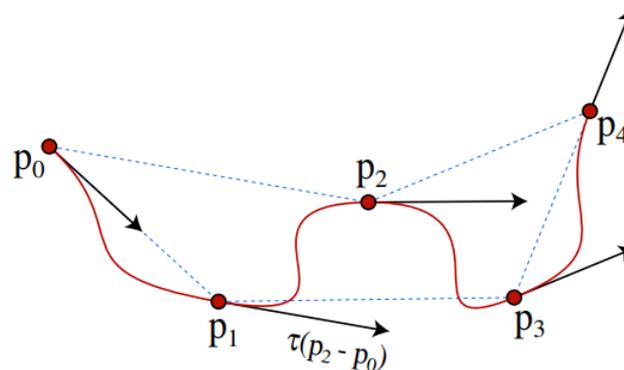
Catmull-Rom Splines

These splines “go through” the points P_0, P_1, P_2, P_3

In matrix representation:

$$\mathbf{P}(u) = \begin{pmatrix} u^3 & u^2 & u & 1 \end{pmatrix} \begin{pmatrix} -\tau & 2-\tau & \tau-2 & \tau \\ 2\tau & \tau-3 & 3-2\tau & -\tau \\ -\tau & 0 & \tau & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \end{pmatrix}$$

With τ being the “tension” or sharpness at the control points (e.g., 0.5)

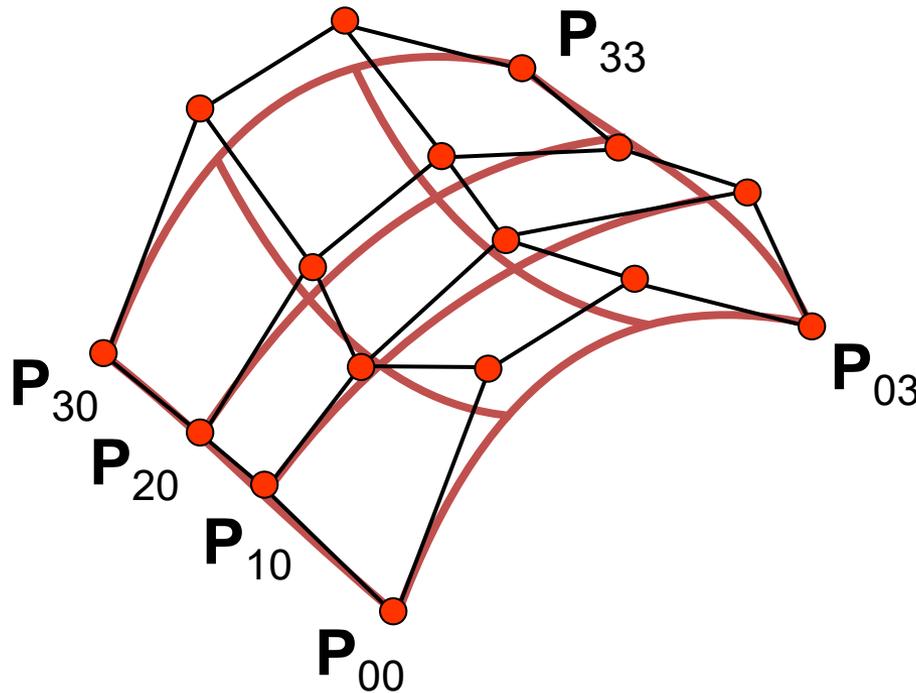




Spline Surface

Control points : 4×4 matrix of points \mathbf{P}_{ij}

Define a smooth patch.



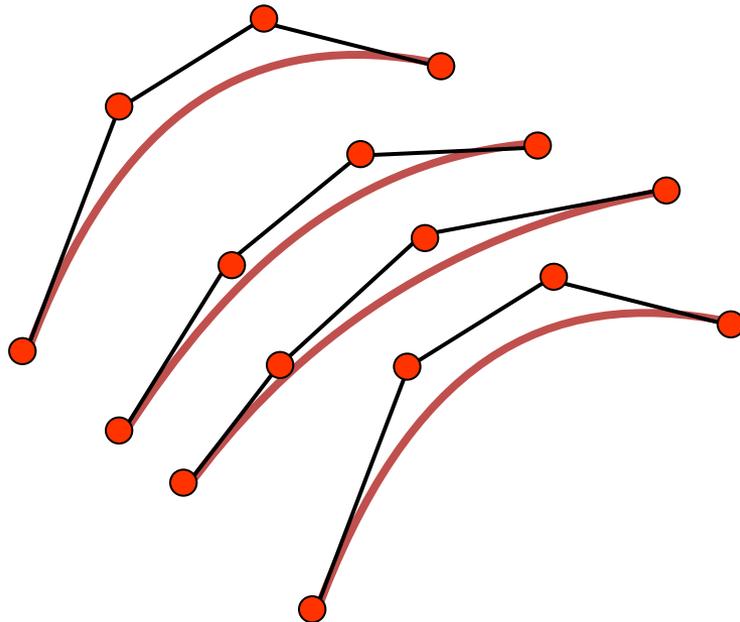


Spline Surface

Control points : 4×4 matrix of points \mathbf{P}_{ij}

Define four curves.

Surface : Smooth these four curves.



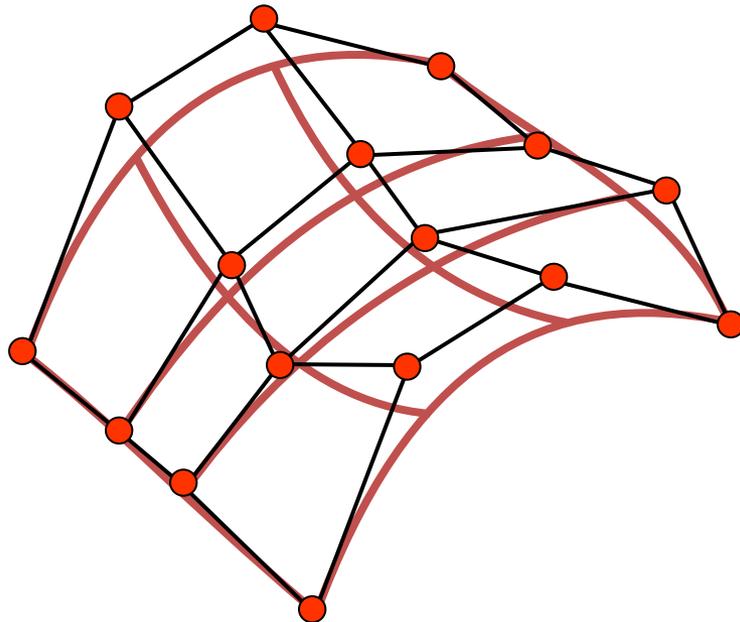


Spline Surface

Control points : 4×4 matrix of points \mathbf{P}_{ij}

Define four curves.

Surface : Smooth these four curves.





Spline Surface

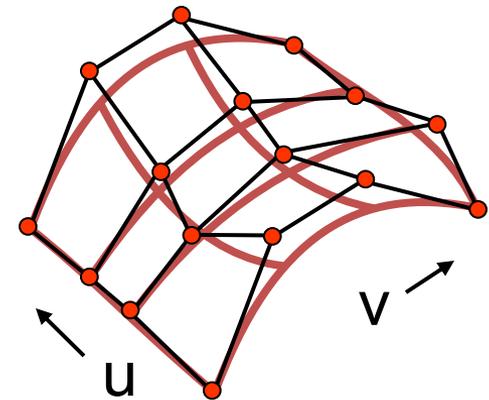
Surface :

Control points : 4×4 matrix of points $\mathbf{P}_{i,j}$

$$\text{Curve : } \mathbf{P}(u) = \sum_{k=0}^3 B_k(u) \mathbf{P}_k$$

$$\text{Surface : } \mathbf{P}(u, v) = \sum_{i=0}^3 B_i(u) \left(\sum_{j=0}^3 B_j(v) \mathbf{P}_{ij} \right)$$

$$= \sum_{i=0}^3 \sum_{j=0}^3 B_i(u) B_j(v) \mathbf{P}_{ij}$$

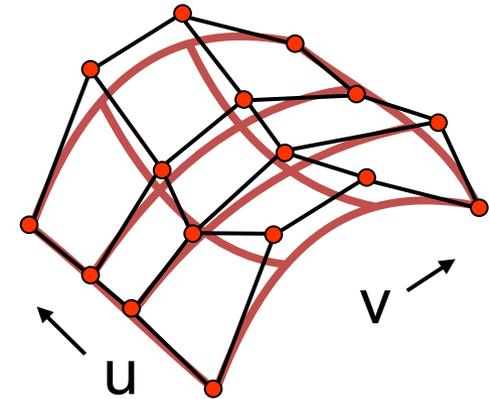




Spline Surface

Display of surface

$$\mathbf{P}(u, v) = \sum_{i=0}^3 \sum_{j=0}^3 B_i(u) B_j(v) \mathbf{P}_{ij}$$



`du := 1/nu; // nu: #facets u-direction`

`dv := 1/nv; // nv: #facets v-direction`

`for i := 0 to nu-1 do`

`u := i*du;`

`for j := 0 to nv - 1 do`

`v := j*dv;`

`DrawQuad(P(u,v), P(u+du, v), P(u+du, v+dv), P(u, v+dv))`



Spline Surface

// Alternative: calculate points first

for i := 0 to nu do

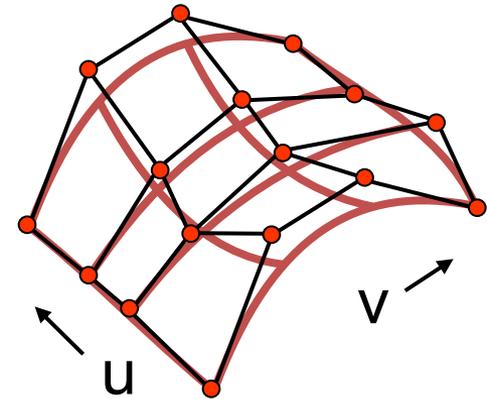
for j := 0 to nv do

$Q[i, j] := P(i/nu, j/nv);$

for i := 0 to nu - 1 do

for j := 0 to nv - 1 do

 DrawQuad($Q[i, j], Q[i+1, j], Q[i+1, j+1], Q[i, j+1]$)



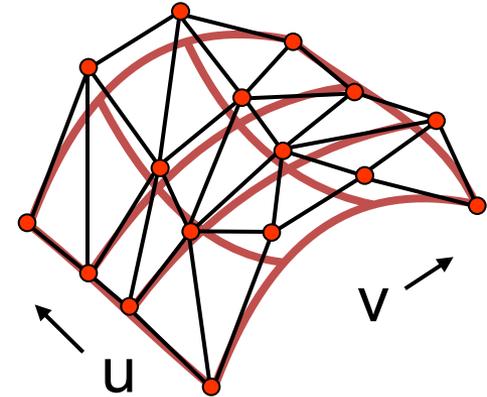


Spline Surface

```
// Alternative: calculate points first,  
// triangle version
```

```
for i := 0 to nu do  
  for j := 0 to nv do  
    Q[i, j] := P(i/nu, j/nv);
```

```
for i := 0 to nu - 1 do  
  for j := 0 to nv - 1 do  
    DrawTriangle(Q[i, j], Q[i+1, j], Q[i+1, j+1]);  
    DrawTriangle(Q[i, j], Q[i+1, j+1], Q[i, j+1]);
```



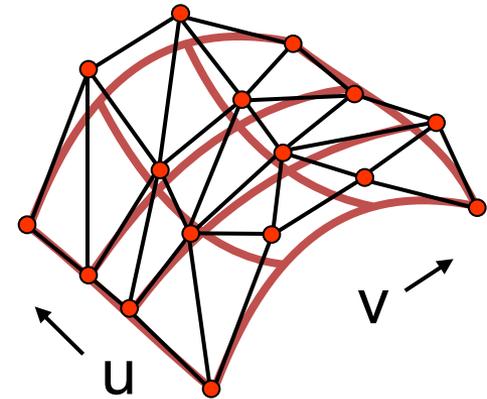


Spline Surface

```
// Alternative: calculate points first,  
// triangle variant, triangle strip
```

```
for i := 0 to nu do  
  for j := 0 to nv do  
    Q[i, j] := P(i/nu, j/nv);
```

```
for i := 0 to nu - 1 do  
  glBegin(GL_TRIANGLE_STRIP);  
  for j := 0 to nv - 1 do  
    glVertex(Q[i, j]);  
    glVertex(Q[i, j+1]);  
  glEnd;
```



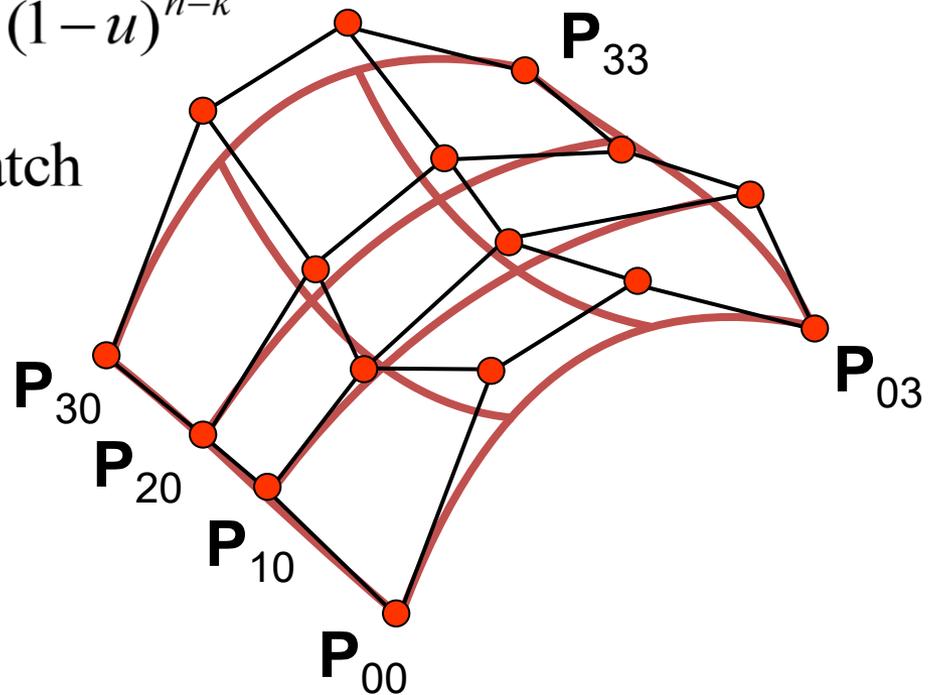


Bézier Surface Patches

$$\text{Generic : } \mathbf{P}(u, v) = \sum_{j=0}^m \sum_{k=0}^n B_j(v) B_k(u) \mathbf{P}_{j,k}$$

$$\text{with } B_{k,n}(u) = \frac{n!}{k!(n-k)!} u^k (1-u)^{n-k}$$

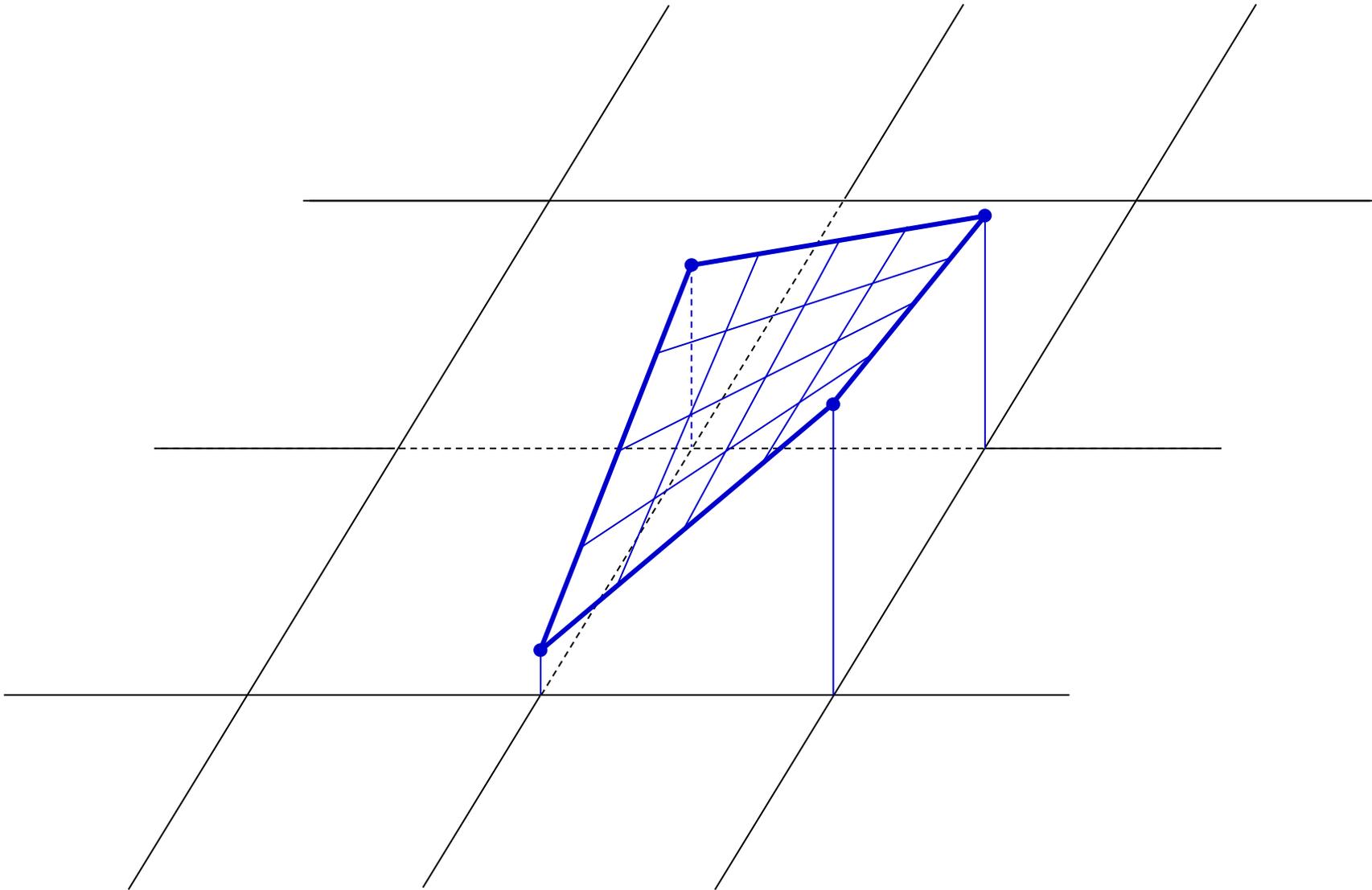
Often $m = n = 4$: bicubic patch



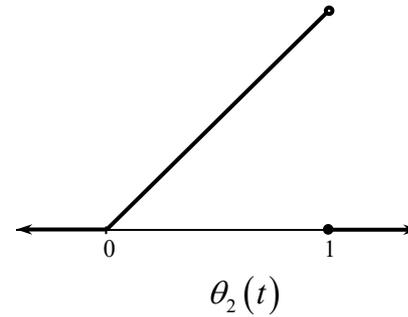
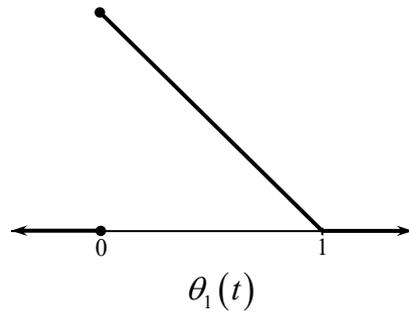
Lets get a bit more complicated...



A closer view of four data points



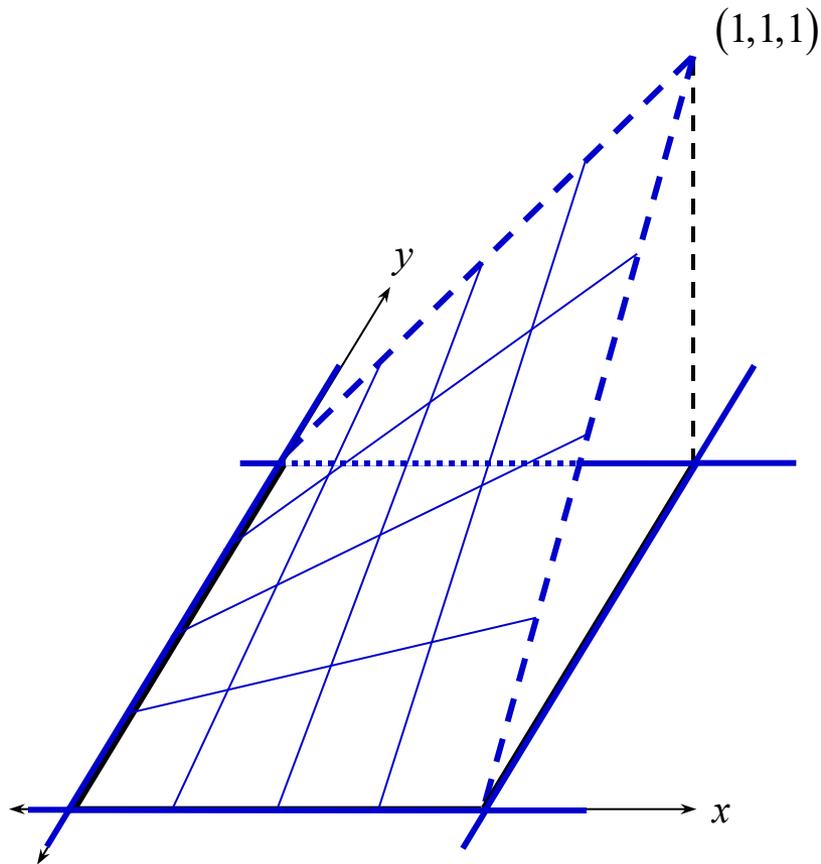
There are two 1-D elementary basis functions



How many 2-D elementary basis functions are there?



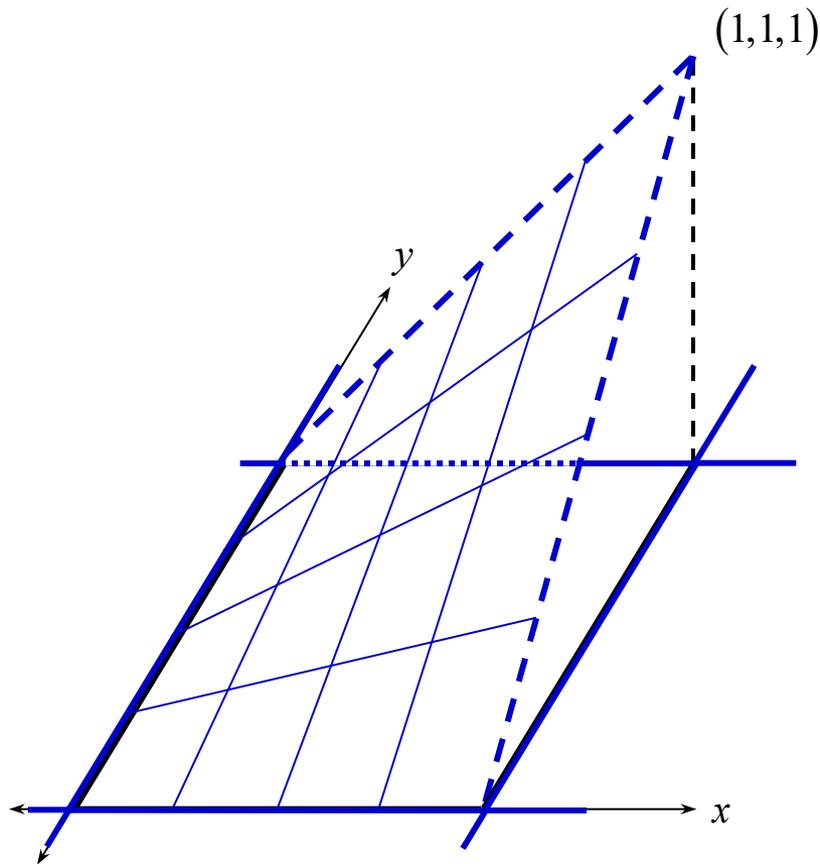
?? Here is one of them:



How many 2-D elementary basis functions are there?



?? Here is one of them:



We define this bilinear 2-D elementary basis function as follows:

$$\theta_{11}(x, y) = \theta_1(x)\theta_1(y)$$

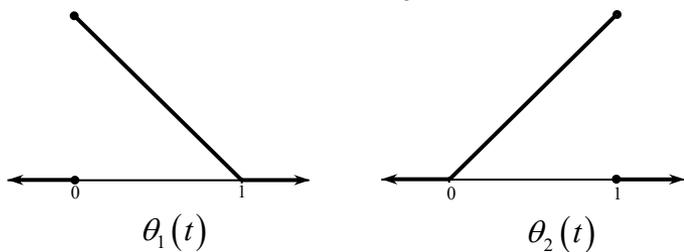
$$\theta_{12}(x, y) = \theta_1(x)\theta_2(y)$$

$$\theta_{21}(x, y) = \theta_2(x)\theta_1(y)$$

$$\theta_{22}(x, y) = \theta_2(x)\theta_2(y)$$

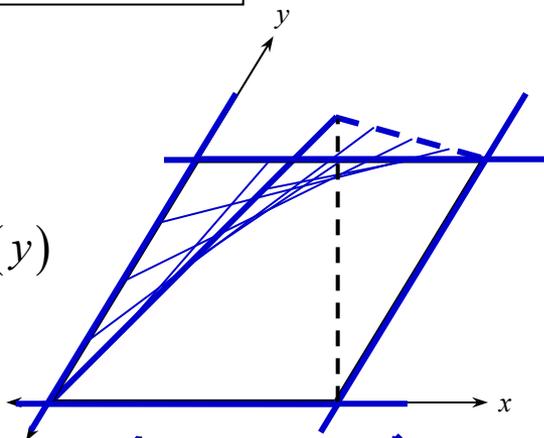


The two 1-D elementary basis functions

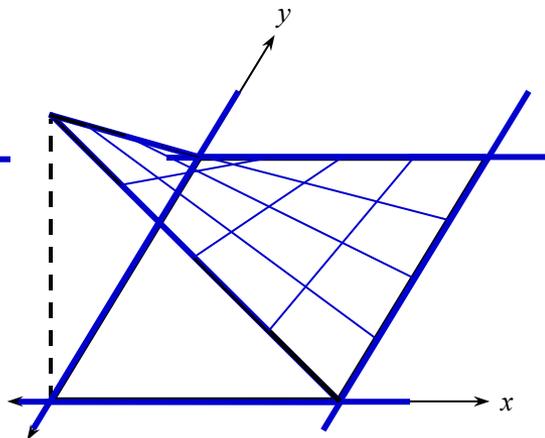


There are four bilinear 2-D elementary basis functions

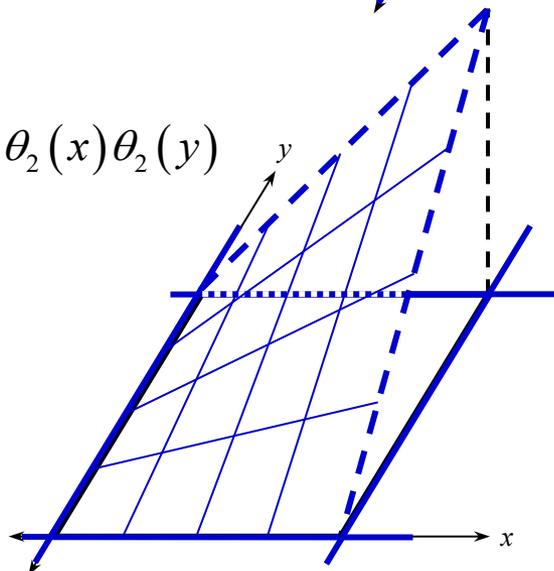
$$\theta_{21}(x, y) = \theta_2(x)\theta_1(y)$$



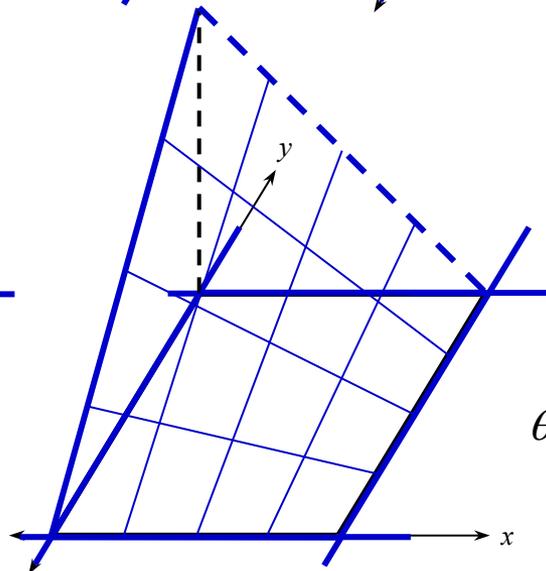
$$\theta_{11}(x, y) = \theta_1(x)\theta_1(y)$$



$$\theta_{22}(x, y) = \theta_2(x)\theta_2(y)$$

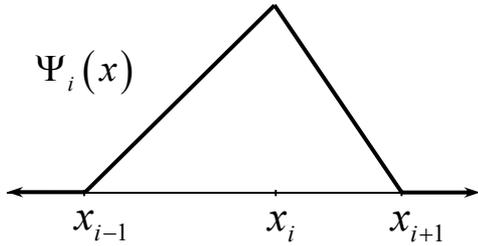


$$\theta_{12}(x, y) = \theta_1(x)\theta_2(y)$$



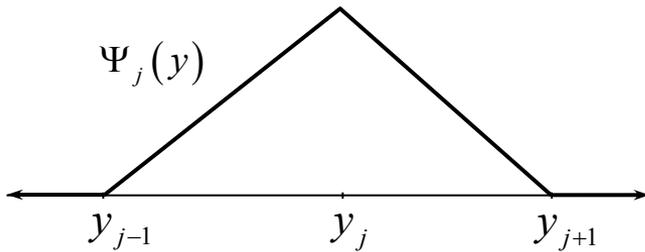


Similarly, we can construct the 2-D bilinear spline basis functions directly from the 1-D linear spline basis functions

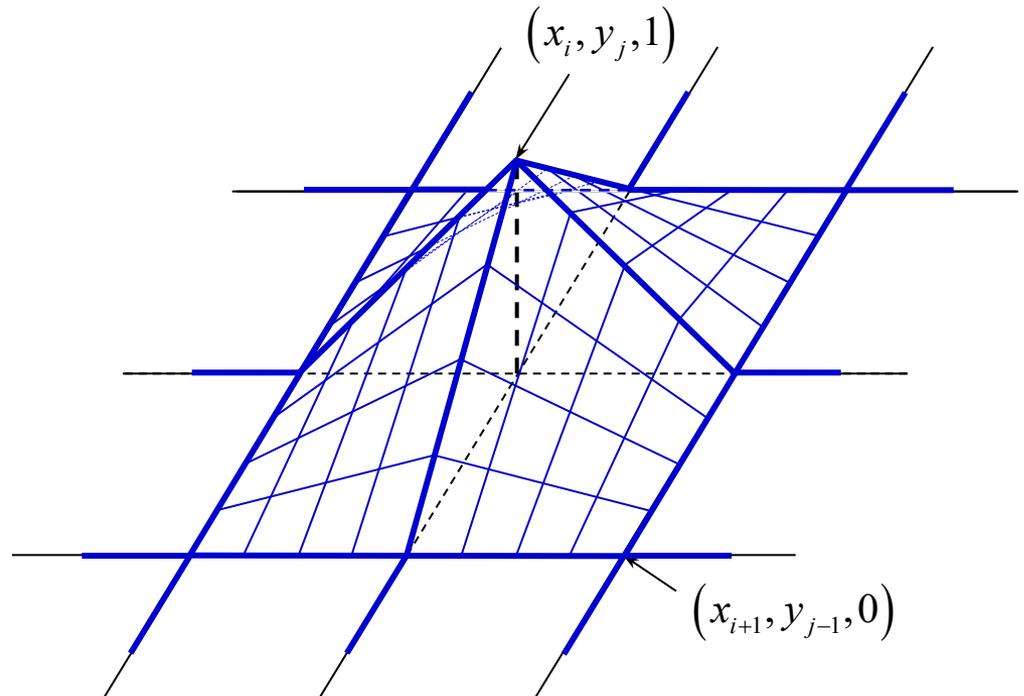


We define the ij th 2-D bilinear spline basis function as follows:

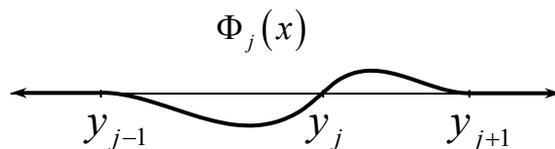
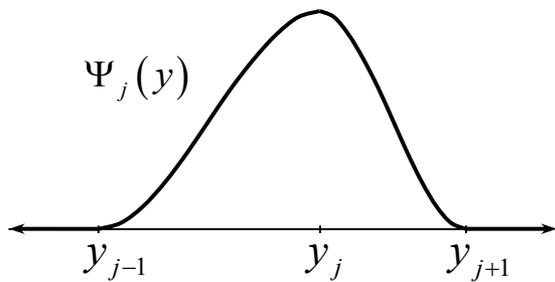
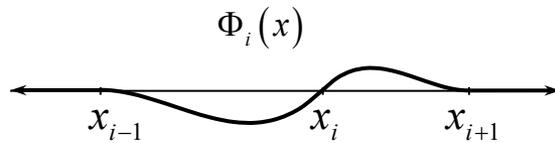
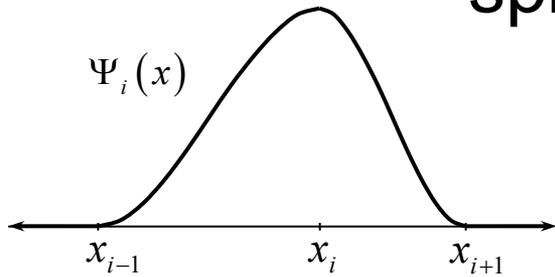
$$\Psi_{ij}(x, y) = \Psi_i(x) \Psi_j(y)$$



As in the 1-D case we want our basis functions to be zero at all but one of the data points.



Similarly, we can construct the 2-D bicubic spline basis functions directly from the 1-D bicubic spline basis functions



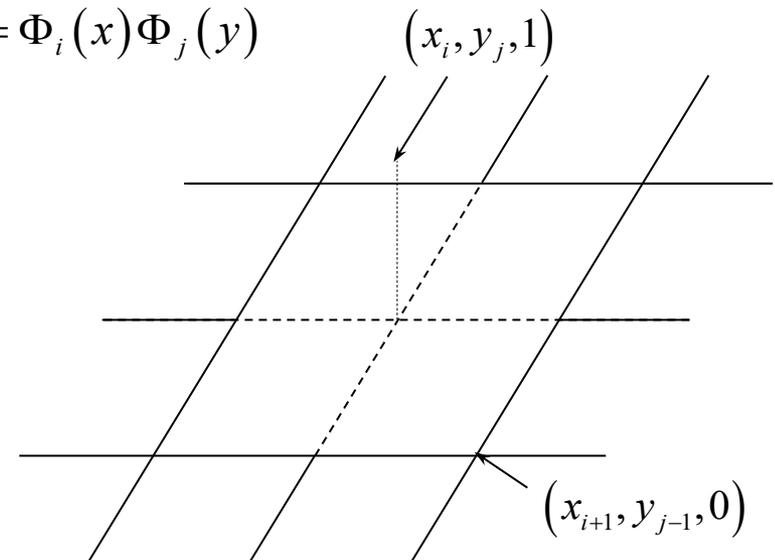
At each data point we define four 2-D bicubic spline basis functions as follows:

$$\Theta^{\Psi\Psi}_{ij}(x, y) = \Psi_i(x)\Psi_j(y)$$

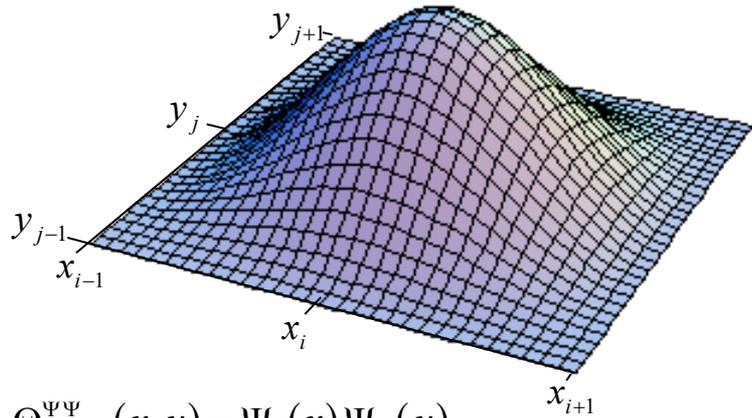
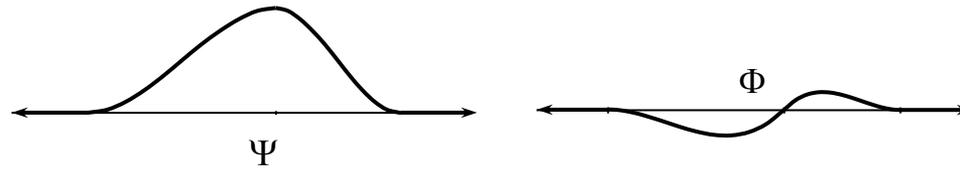
$$\Theta^{\Psi\Phi}_{ij}(x, y) = \Psi_i(x)\Phi_j(y)$$

$$\Theta^{\Phi\Psi}_{ij}(x, y) = \Phi_i(x)\Psi_j(y)$$

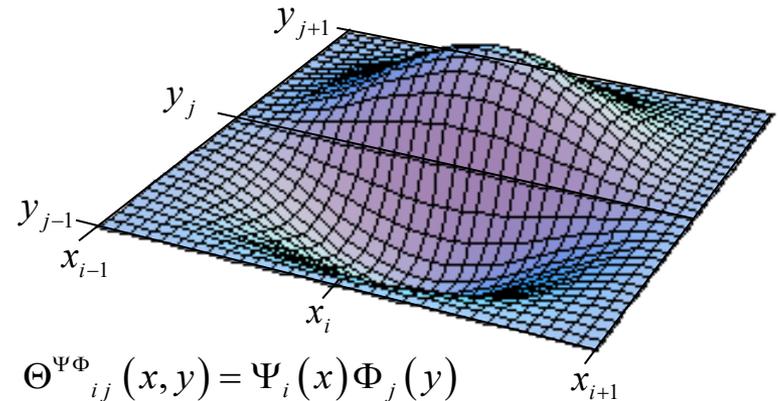
$$\Theta^{\Phi\Phi}_{ij}(x, y) = \Phi_i(x)\Phi_j(y)$$



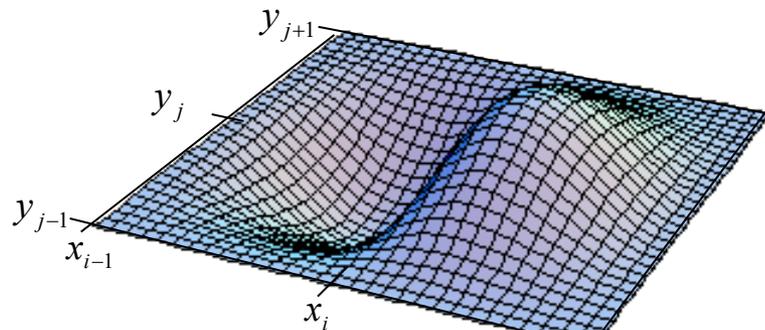
The 2-D bicubic spline basis functions



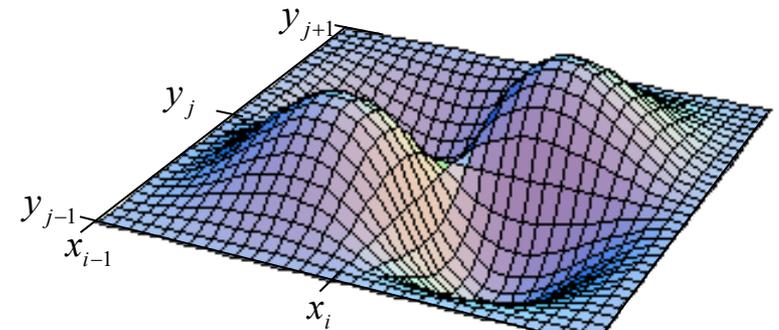
$$\Theta^{\Psi\Psi}_{ij}(x, y) = \Psi_i(x)\Psi_j(y)$$



$$\Theta^{\Psi\Phi}_{ij}(x, y) = \Psi_i(x)\Phi_j(y)$$



$$\Theta^{\Phi\Psi}_{ij}(x, y) = \Phi_i(x)\Psi_j(y)$$



$$\Theta^{\Phi\Phi}_{ij}(x, y) = \Phi_i(x)\Phi_j(y)$$



Thus, what can we do?

- With 2D elementary basis functions?
- With 2D bilinear spline basis?
- With 2D bicubic basis functions?

Lots! Welcome to the world of Basis functions!



- Radial Basis Ftns:

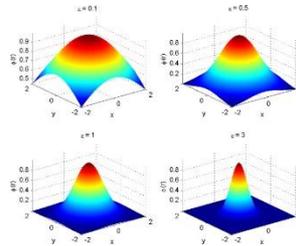
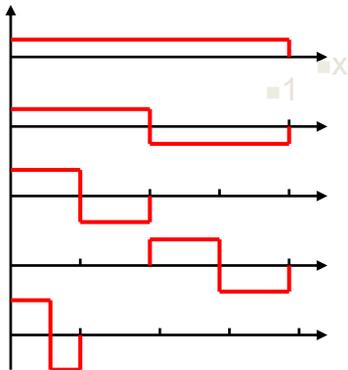
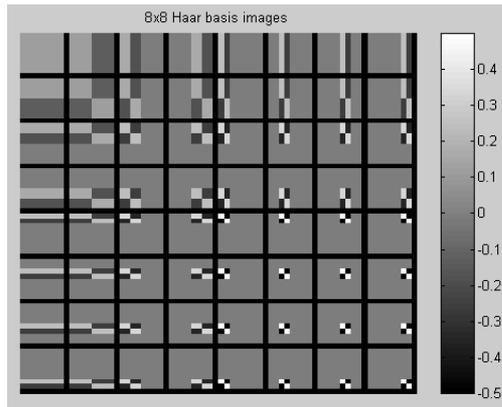


Figure 1: (a) Fitting a Radial Basis Function (RBF) to a 438,000 point-cloud. (b) Automatic mesh repair using the biharmonic RBF.

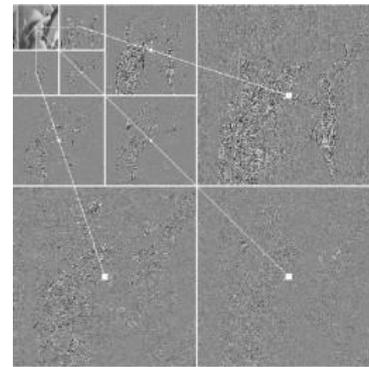
- Haar Basis (Wavelets):



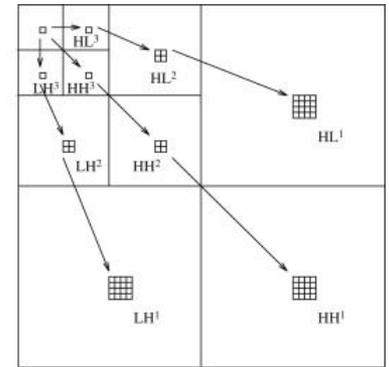
1D



2D



(a)



(b)

Image compression