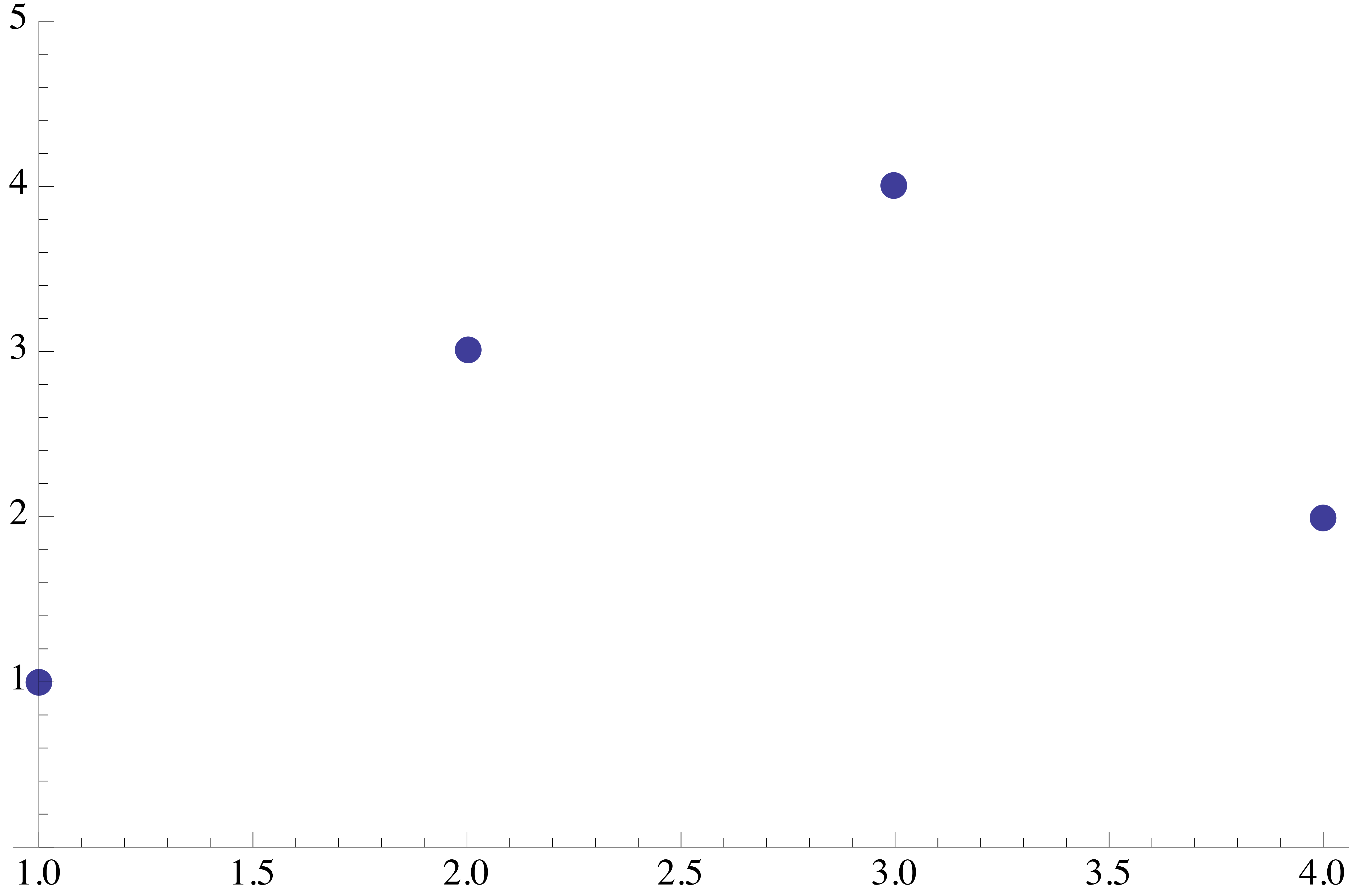
Interpolation

CS 355: Introduction to Graphics and Image Processing

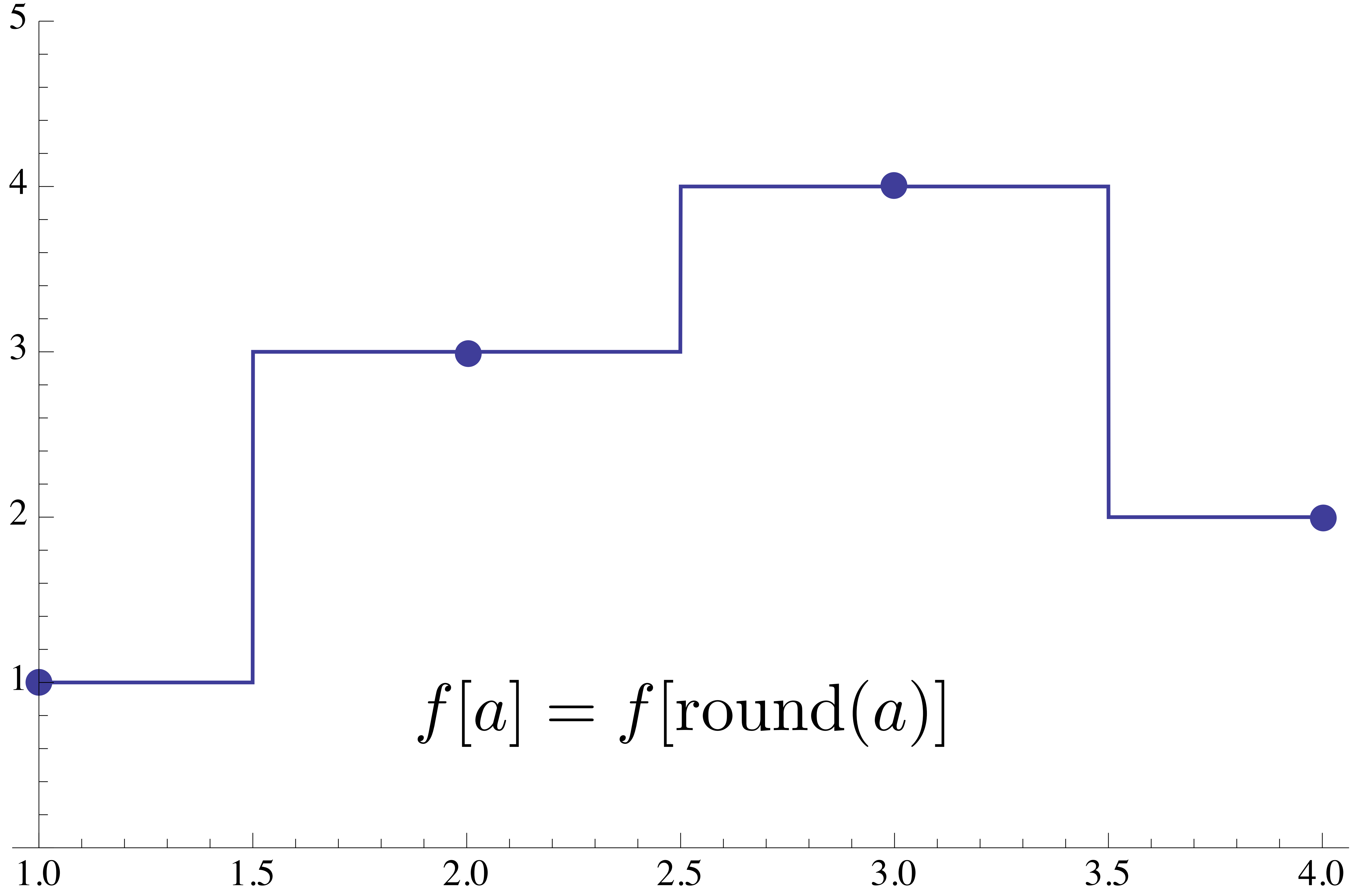
Interpolation

- Used to estimate a function between known sampled values
- Digital signals to analog
  - Audio playback
  - Image display
- Spatial image manipulation
  - Changing size
  - Rotation
  - Warping
- Generate smooth geometric models between defined points (we'll come back to this later)

Interpolation



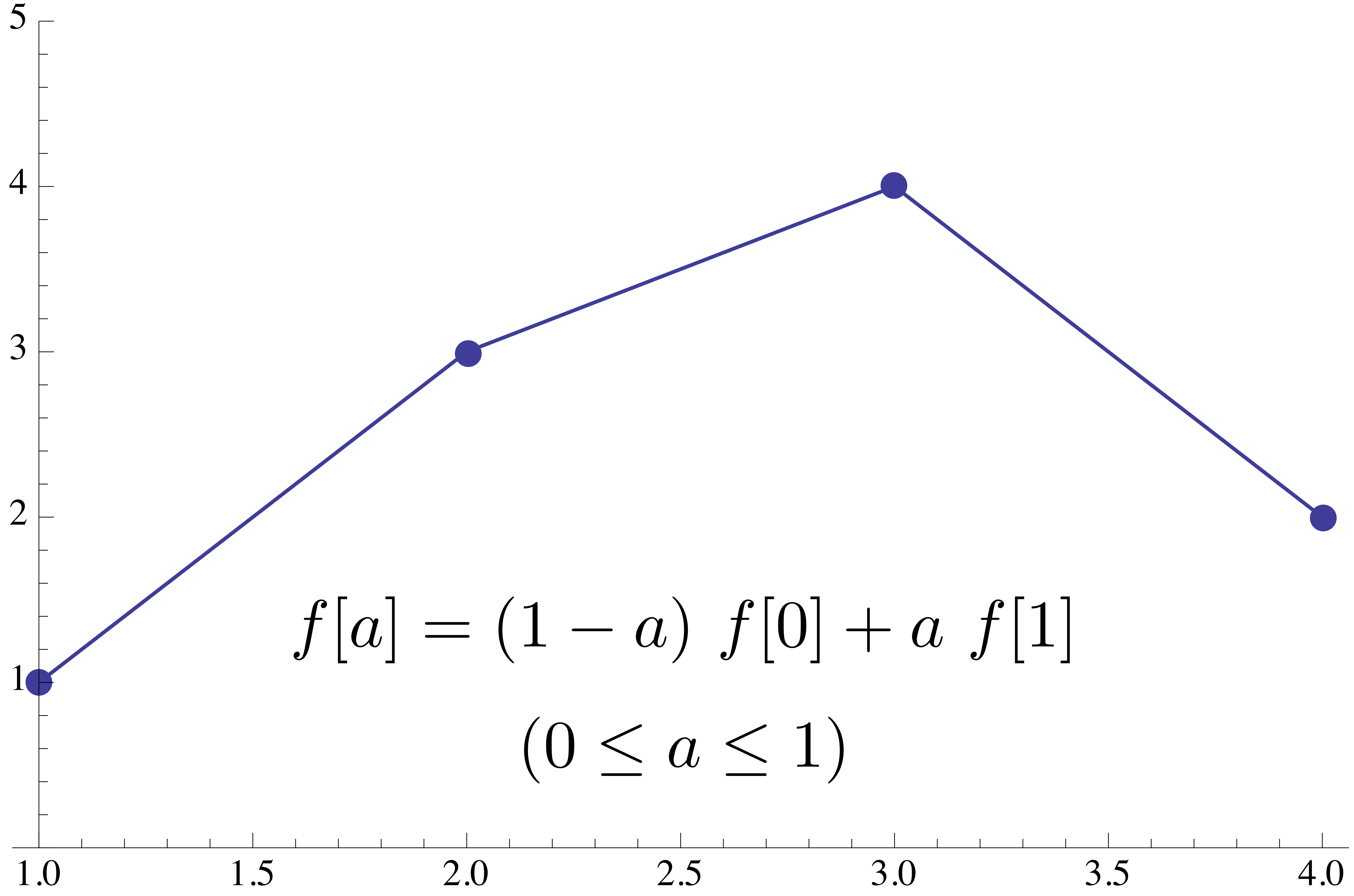
Nearest Neighbor



$$f[a] = f[\mathrm{round}(a)]$$

0th order

Linear Interpolation

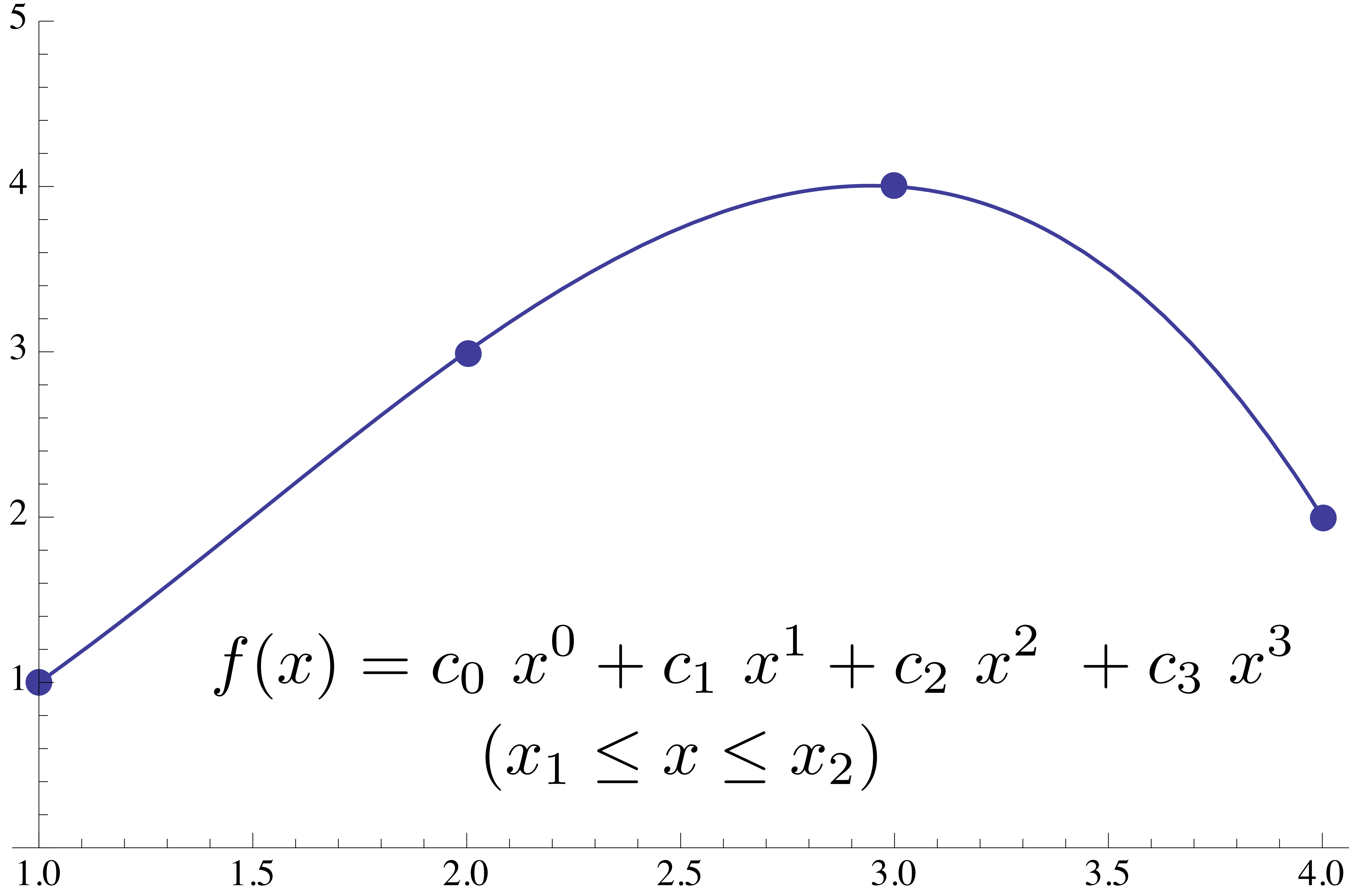


$$f[a] = (1 - a)\ f[0] + a\ f[1]$$

$$(0 \leq a \leq 1)$$

1st order

Cubic Interpolation



$$f(x) = c_0\ x^0 + c_1\ x^1 + c_2\ x^2\ + c_3\ x^3$$
$$(x_1 \leq x \leq x_2)$$

3rd order

Cubic Interpolation

$$f(x) = ax^3 + bx^2 + cx + d$$

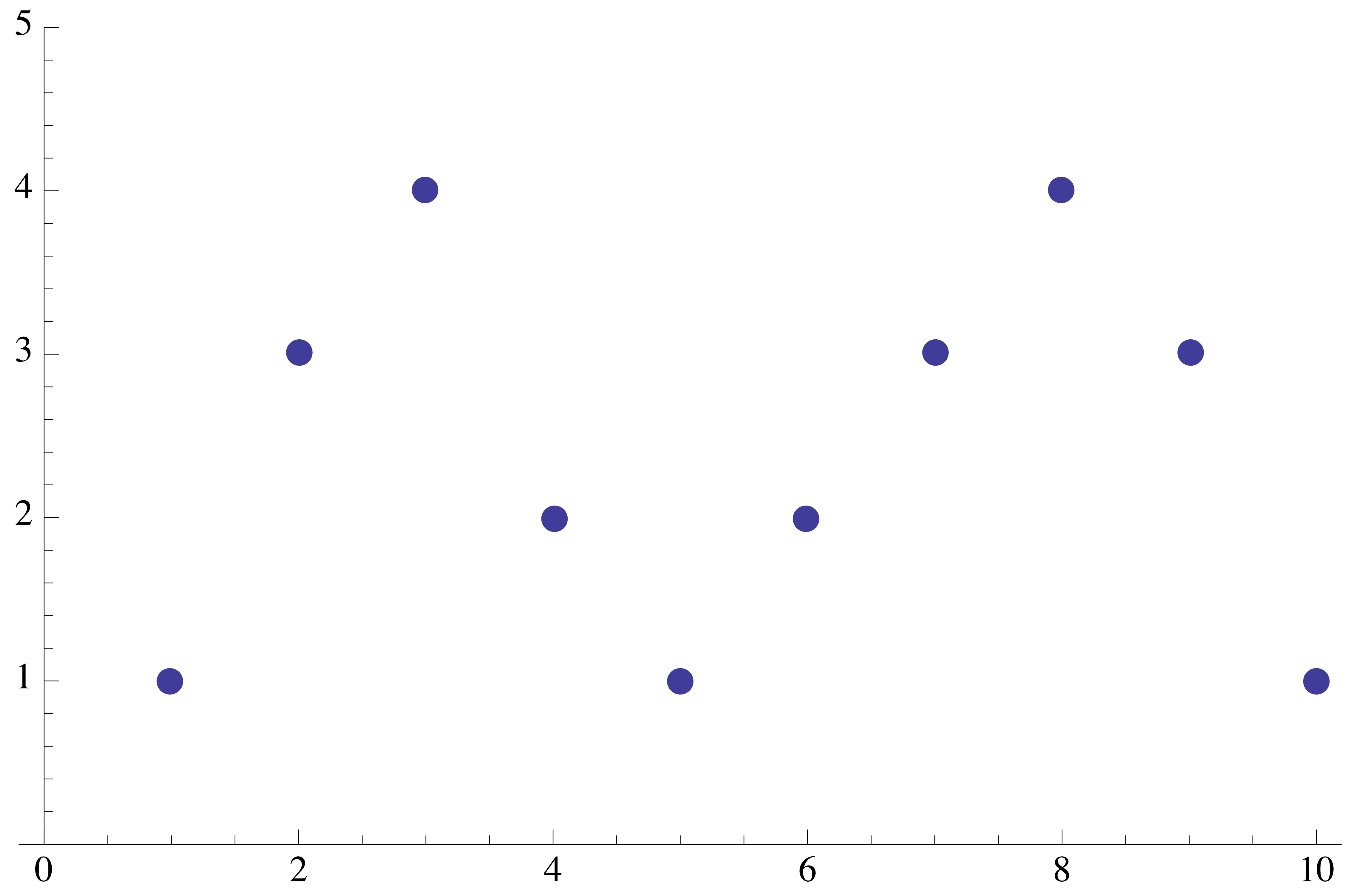
$$(x_1 \leq x \leq x_2)$$

$$\begin{aligned} f(x_0) &= a\ x_0^3 + b\ x_0^2 + c\ x_0 + d \\ f(x_1) &= a\ x_1^3 + b\ x_1^2 + c\ x_1 + d \\ f(x_2) &= a\ x_2^3 + b\ x_2^2 + c\ x_2 + d \\ f(x_3) &= a\ x_3^3 + b\ x_3^2 + c\ x_3 + d \end{aligned}$$

$$\begin{bmatrix} x_0^0 & x_0^1 & x_0^2 & x_0^3 \\ x_1^0 & x_1^1 & x_1^2 & x_1^3 \\ x_2^0 & x_2^1 & x_2^2 & x_2^3 \\ x_3^0 & x_3^1 & x_3^2 & x_3^3 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} f(x_0) \\ f(x_1) \\ f(x_2) \\ f(x_3) \end{bmatrix}$$

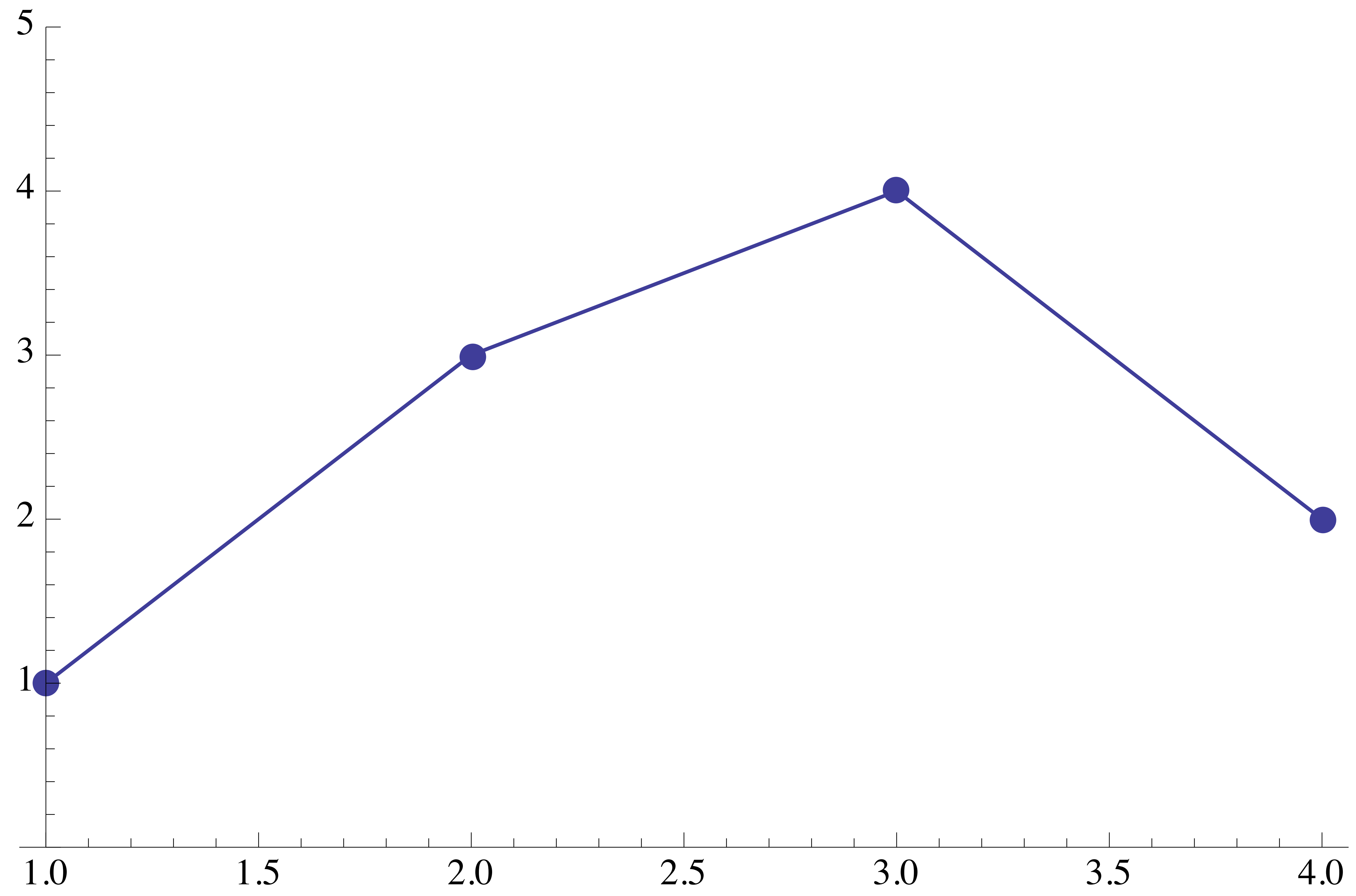
Solve for a, b, c, d and plug into function

Longer Sequences



0th - nearest neighbor
1st - use one sample on both sides
3rd - use two samples on both sides

Continuity



0th order: function is continuous
1st order: slope is continuous
nth order: nth derivative is continuous

2D Images

- Extend ideas of single-value interpolation
    - Nearest neighbor
    - Linear
    - Cubic
    - …

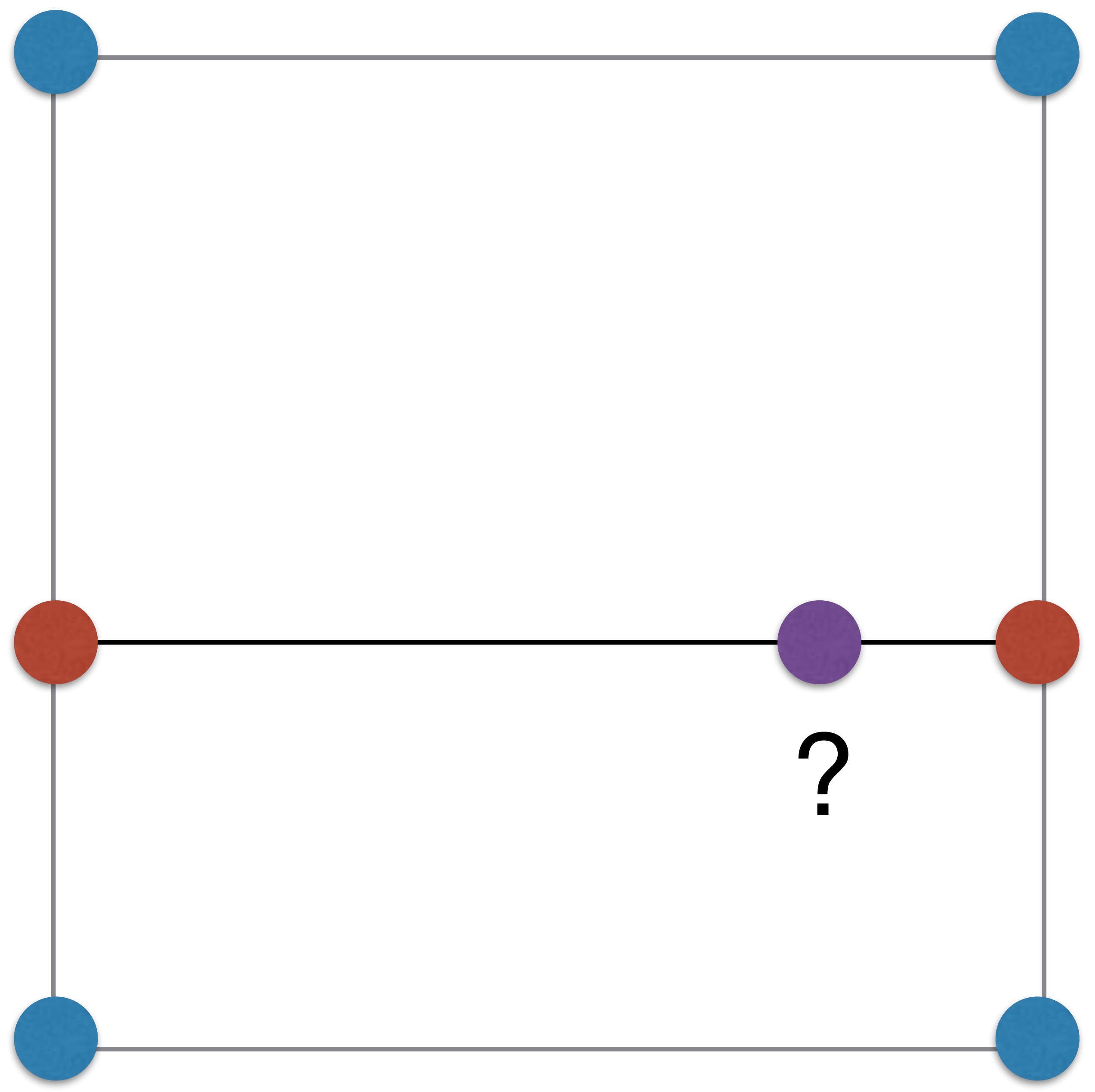
Nearest Neighbor

- Same idea as in 1-d:
  Round off to nearest pixel
- Also called “pixel replication”
- Big blocky pixels

Bilinear

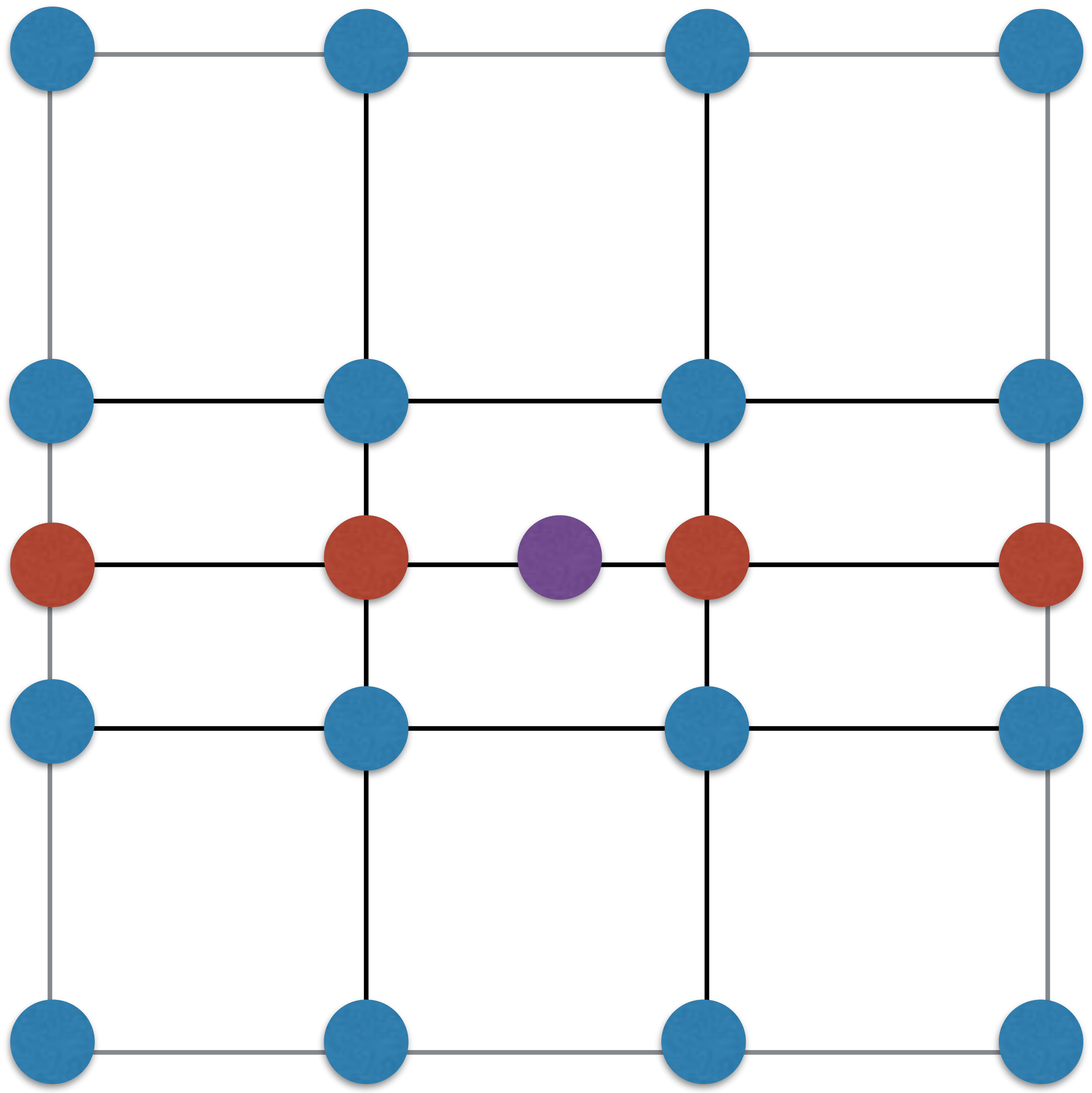
- Very common approach:
  - Interpolate vertically
  - Interpolate horizontally

  (or vice versa)
- For linear interpolation, this is called *bilinear*



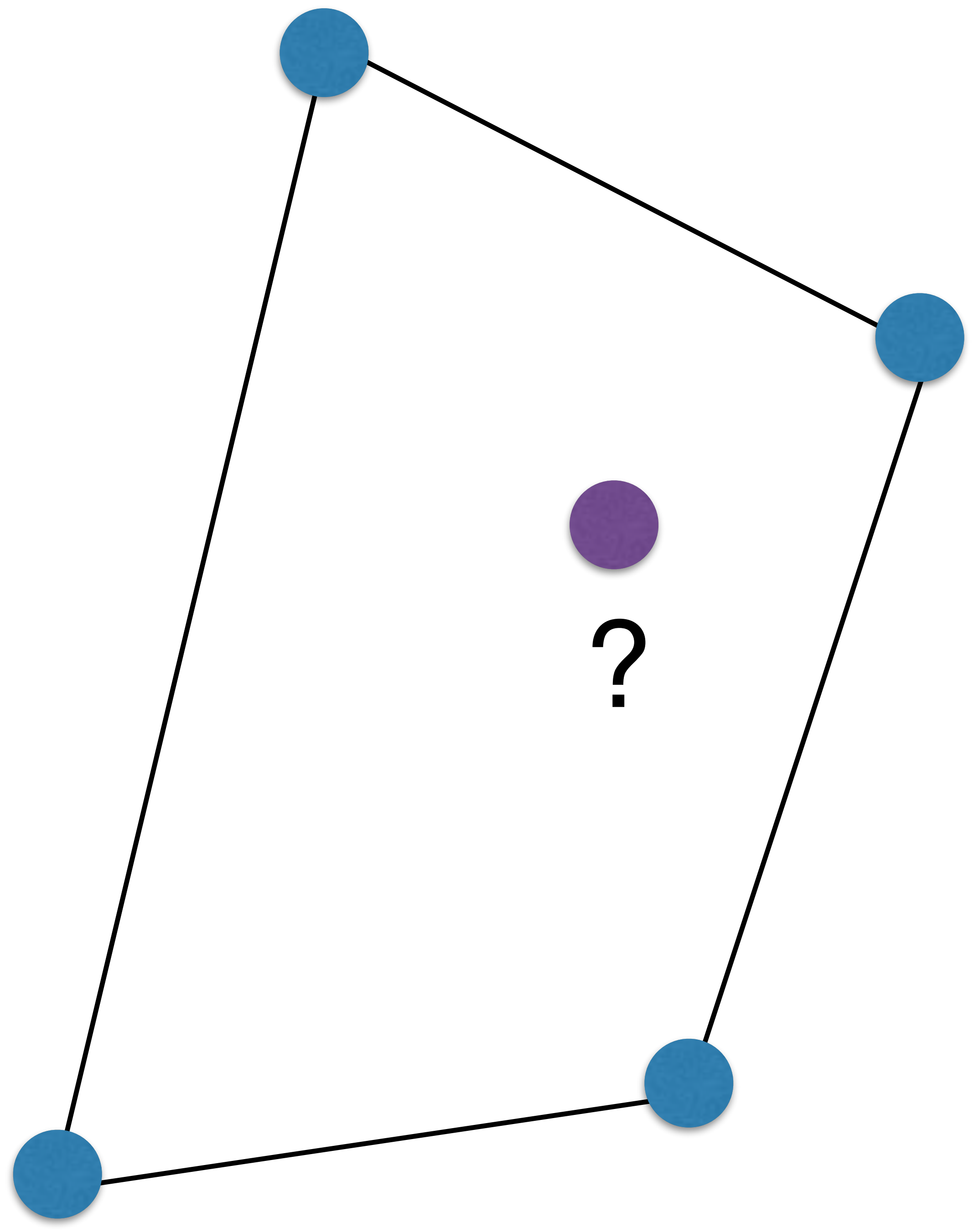
Bicubic

- Same idea as bilinear but using a 4 x 4 grid of neighbors
- Tends to produce sharper results
- More computationally intensive



Generalizing Bilinear

- Corner points don't have to lie on a square or rectangle
- Can be any quadrilateral



Generalized Bilinear

- General form:

$$f(x,y) = ax + by + cxy + d$$

- Use similar strategy as with curve fitting:

$$\begin{aligned}
f(x_1, y_1) &= ax_1 + by_1 + cx_1y_1 + d \\
f(x_2, y_2) &= ax_2 + by_2 + cx_2y_2 + d \\
f(x_3, y_3) &= ax_3 + by_3 + cx_3y_3 + d \\
f(x_4, y_4) &= ax_4 + by_4 + cx_4y_4 + d
\end{aligned}$$

Solve for a, b, c, d and plug into function

Coming up…

- Image warping
- Texture mapping