Spatial Data

CS444

Chapter 8, VA&D

http://www.sci.utah.edu/~miriah/cs6630/lectures/L17-isosurfaces.pdf

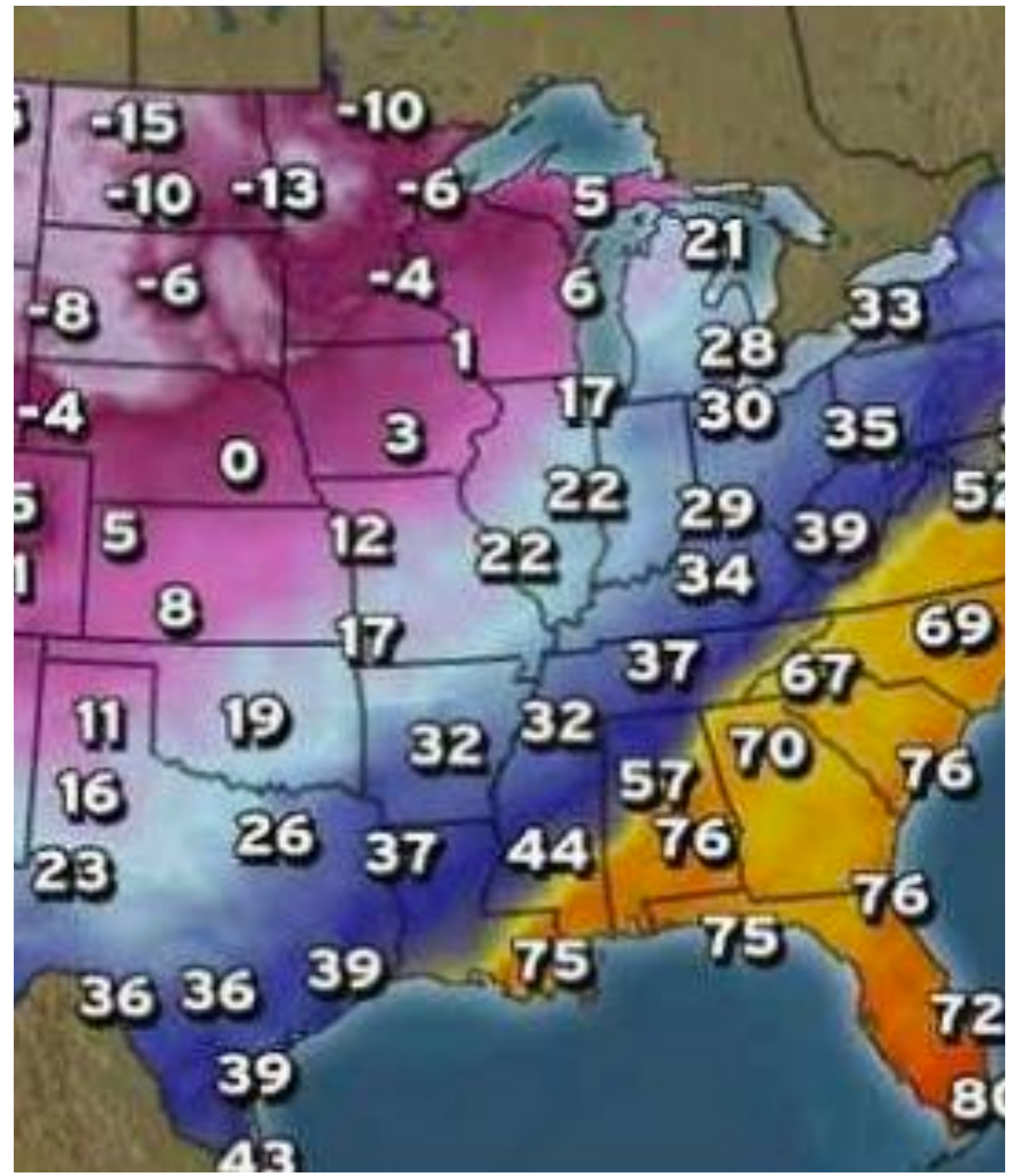
http://www.slate.com/blogs/future_tense/2013/12/06/winter_storm_cleon_record_lows_us_weather_map_today_is_completely_insane.html

Recap

- So far, we've studied methods to determine the position of a data point on the screen
    - graph drawing, treemaps, scatterplots, PCA
- However, some datasets come with very good positional information
    - Wind maps, weather simulations, CT scans

How do we represent spatial data?

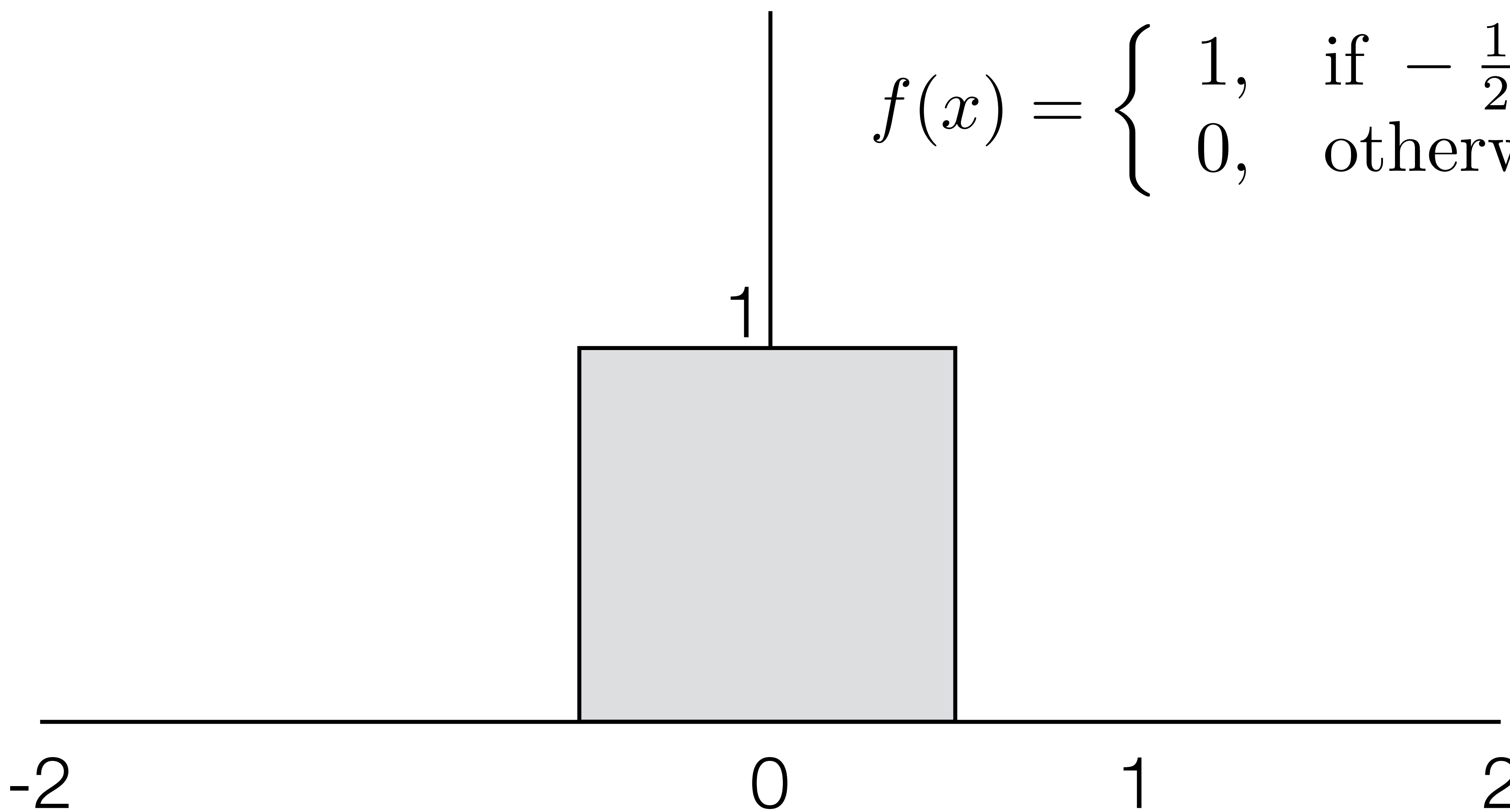
- In the real world, there's infinitely many data points in a weather map
- In a computer, we only have finite memory and finite time
- How do we solve this problem?



Finite-dimensional function spaces

- **Some** functions can be represented succinctly

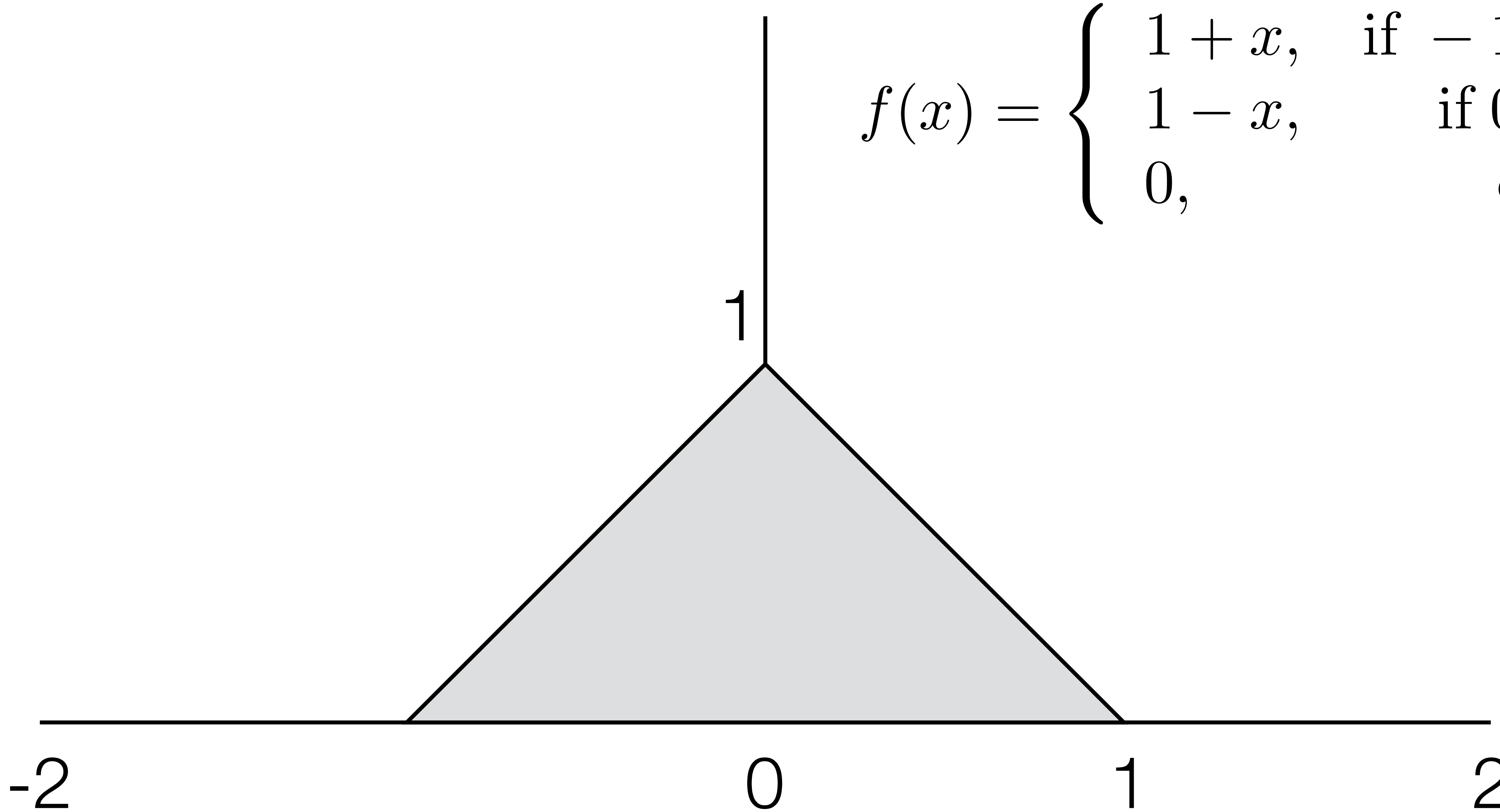
$$f(x) = \begin{cases} 1, & \text{if } -\frac{1}{2} \leq x \leq \frac{1}{2} \\ 0, & \text{otherwise} \end{cases}$$



Finite-dimensional function spaces

- **Some** functions can be represented succinctly

$$f(x) = \begin{cases} 1 + x, & \text{if } -1 \le x \le 0 \\ 1 - x, & \text{if } 0 \le x \le 1 \\ 0, & \text{otherwise} \end{cases}$$

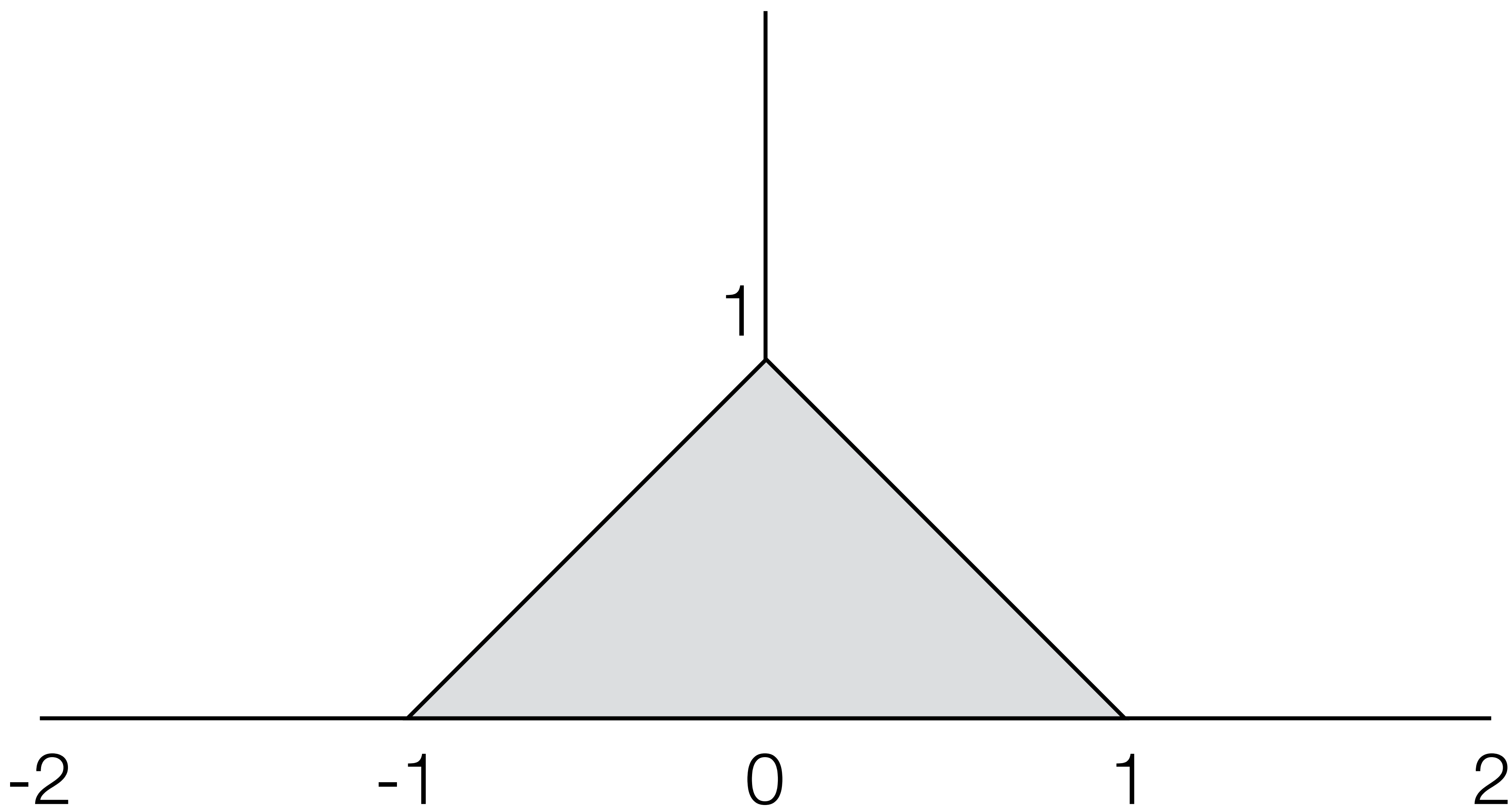


Finite-dimensional function spaces

- Build complicated functions by **sums** of **shifted**, **scaled** versions of these simple functions

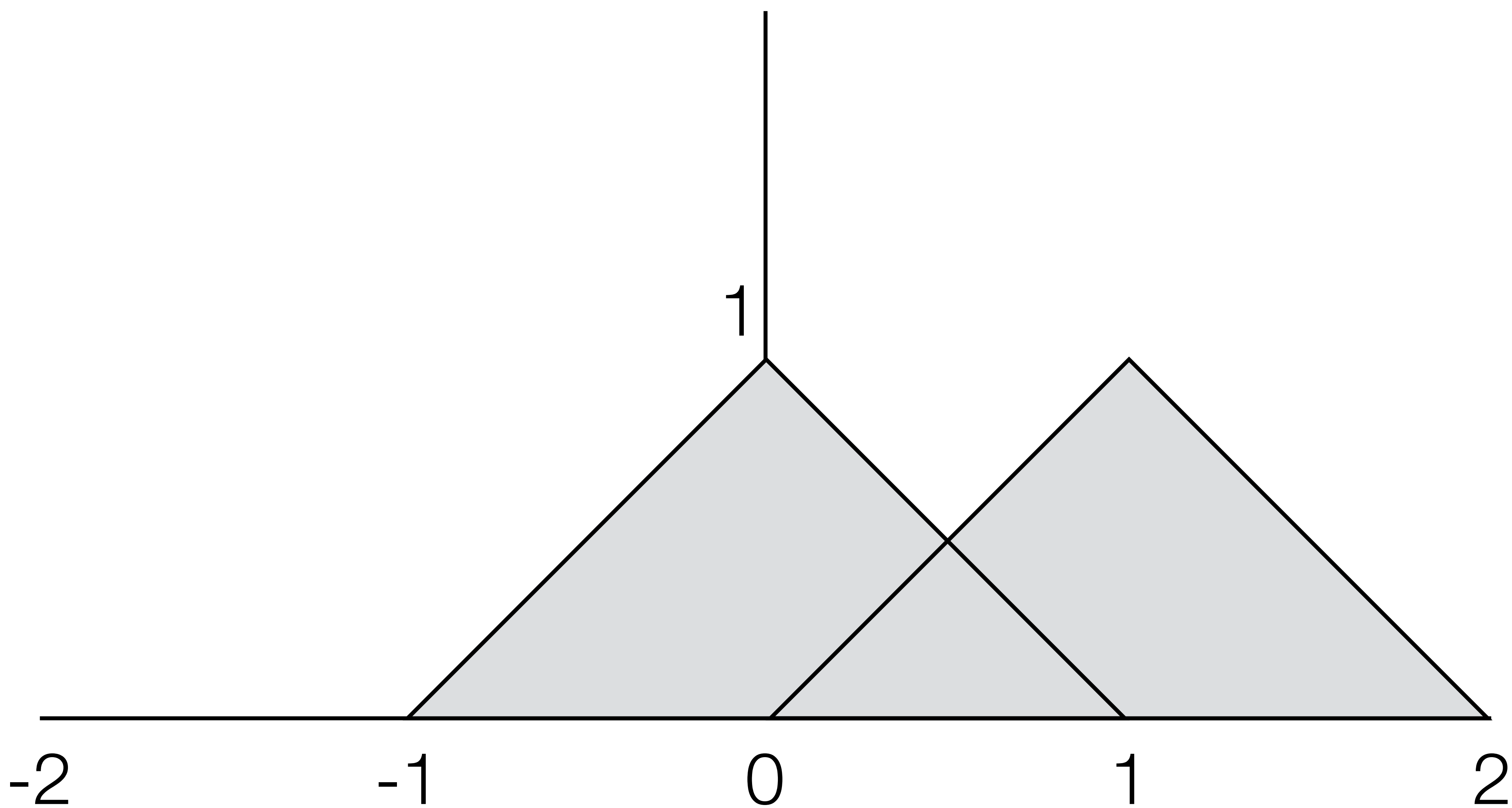
Finite-dimensional function spaces

- Build complicated functions by **sums** of **shifted**, **scaled** versions of these simple functions



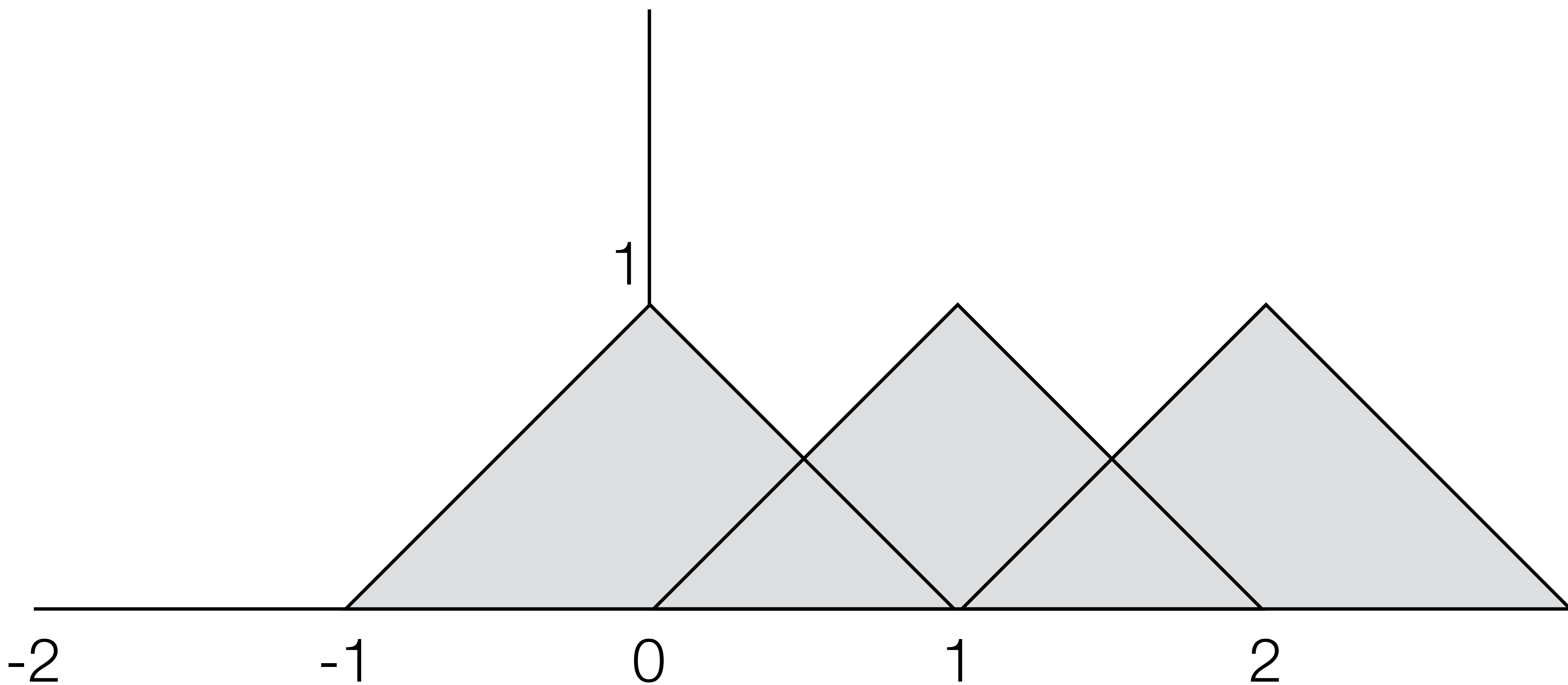
Finite-dimensional function spaces

- Build complicated functions by **sums** of **shifted**, **scaled** versions of these simple functions



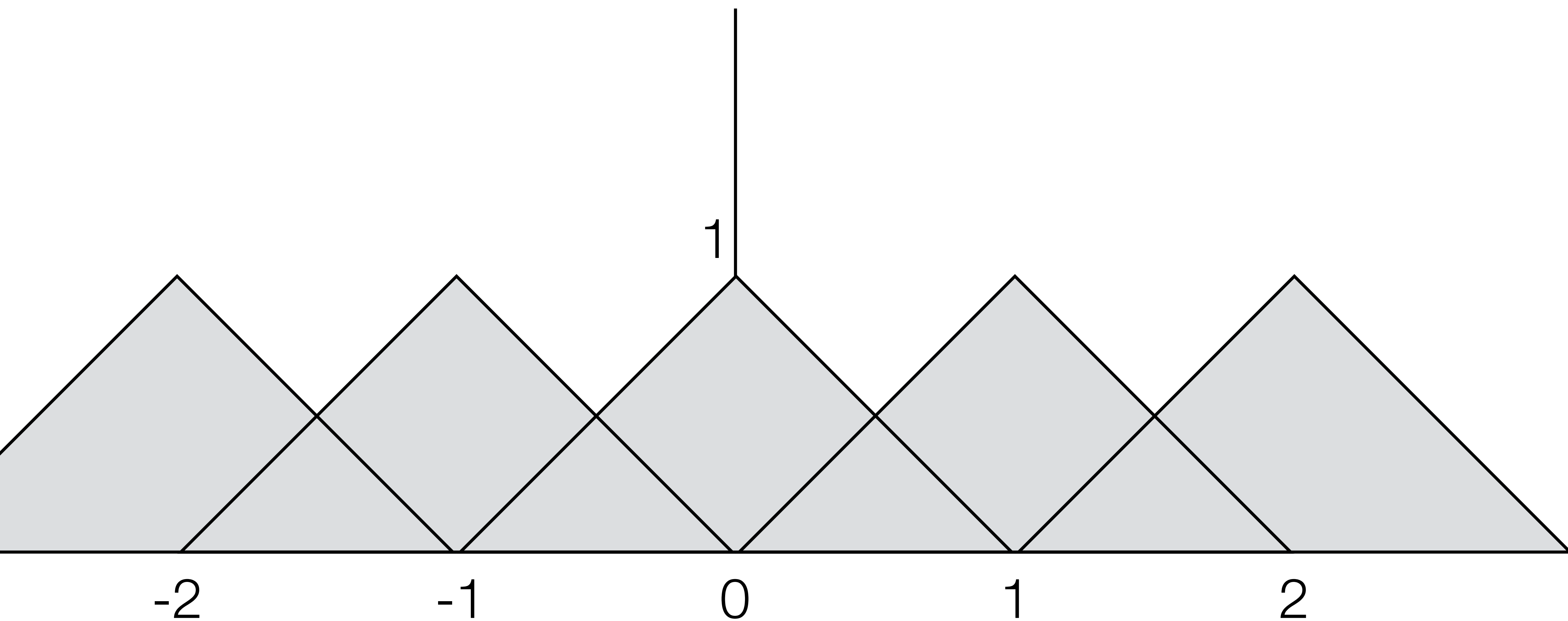
Finite-dimensional function spaces

- Build complicated functions by **sums** of **shifted**, **scaled** versions of these simple functions



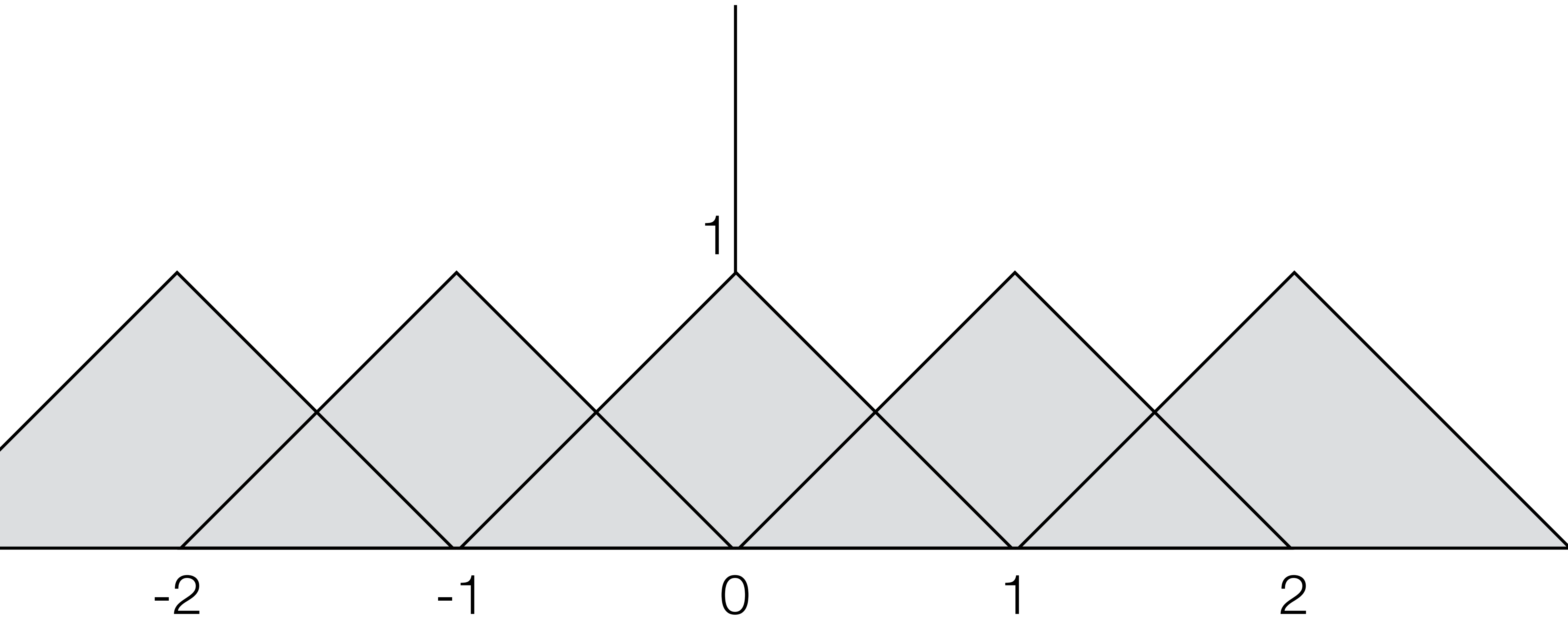
Finite-dimensional function spaces

- Build complicated functions by **sums** of **shifted**, **scaled** versions of these simple functions



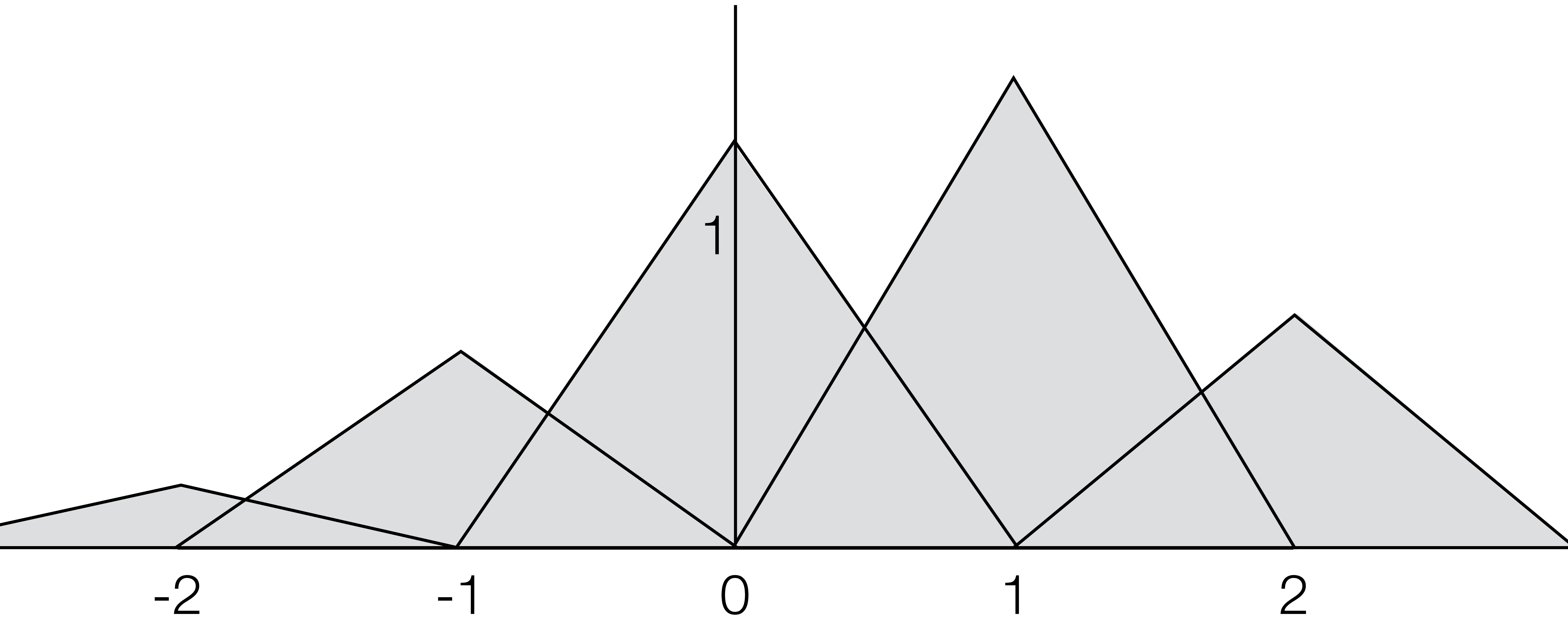
Finite-dimensional function spaces

- Build complicated functions by **sums** of **shifted**, **scaled** versions of these simple functions



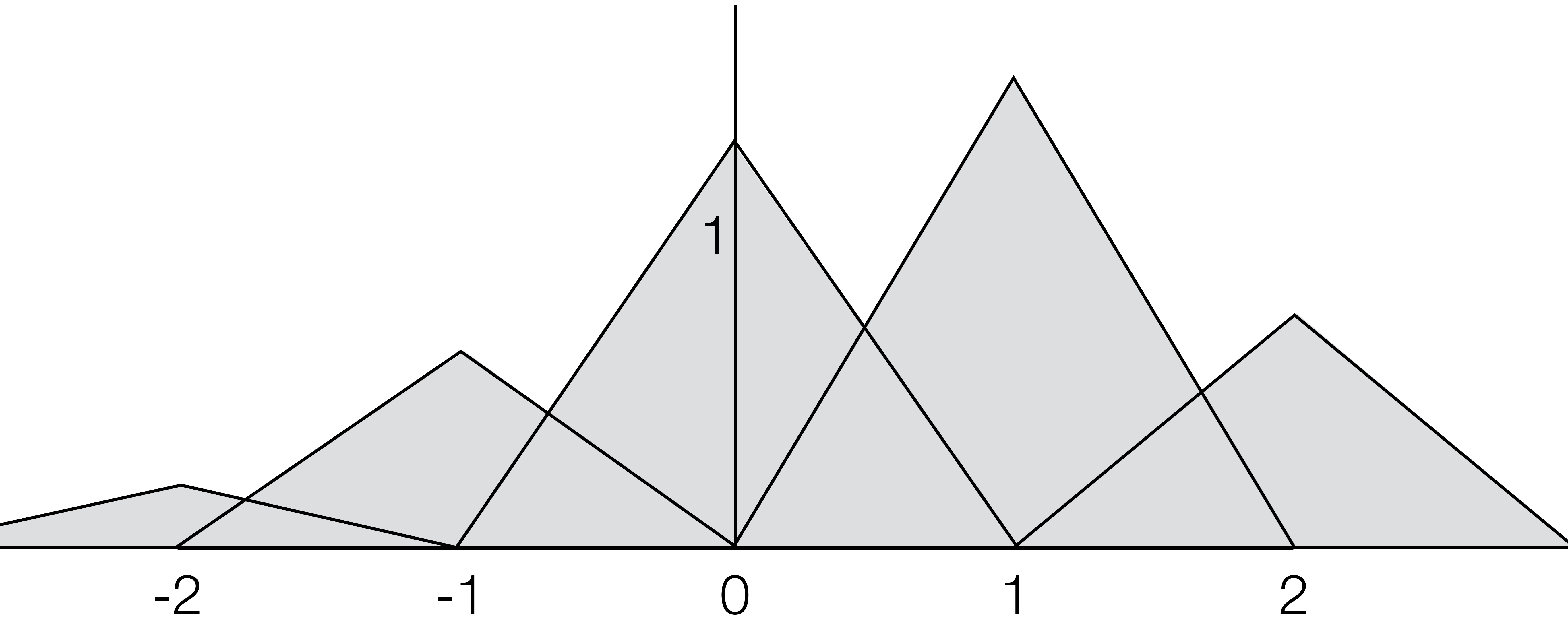
Finite-dimensional function spaces

- Build complicated functions by **sums** of **shifted**, **scaled** versions of these simple functions



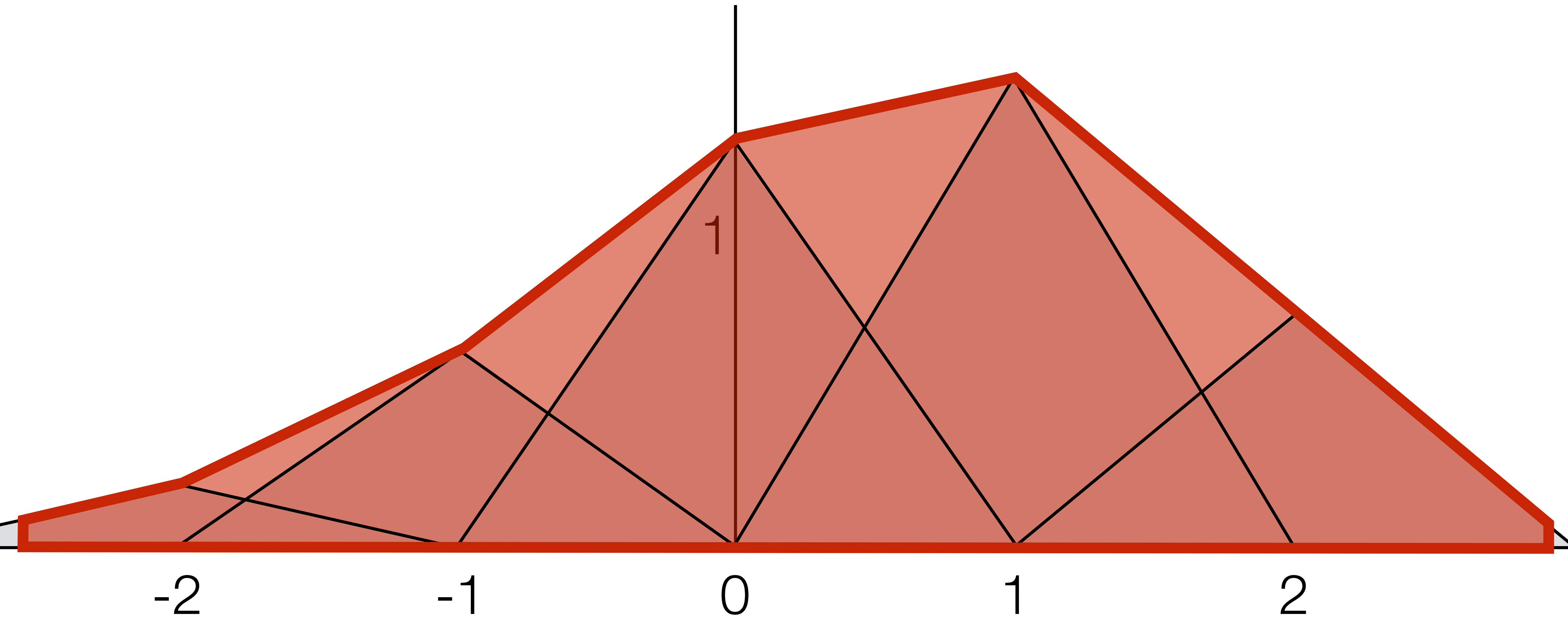
Finite-dimensional function spaces

- Build complicated functions by **sums** of **shifted**, **scaled** versions of these simple functions

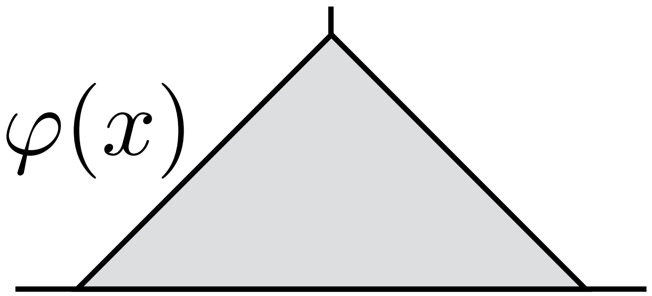


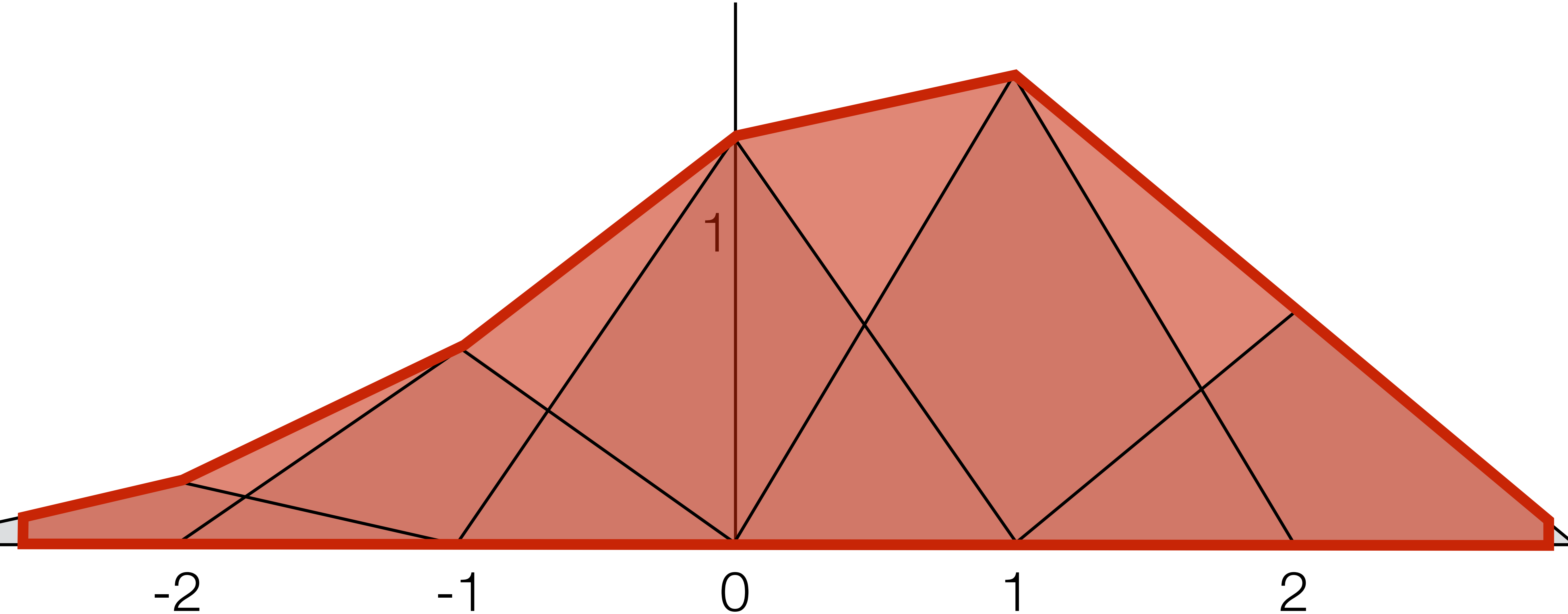
Finite-dimensional function spaces

- Build complicated functions by **sums** of **shifted**, **scaled** versions of these simple functions



$$f(x) = \sum_i \varphi(x - i)c_i$$

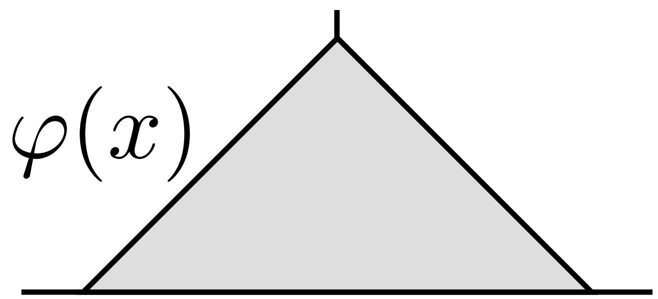


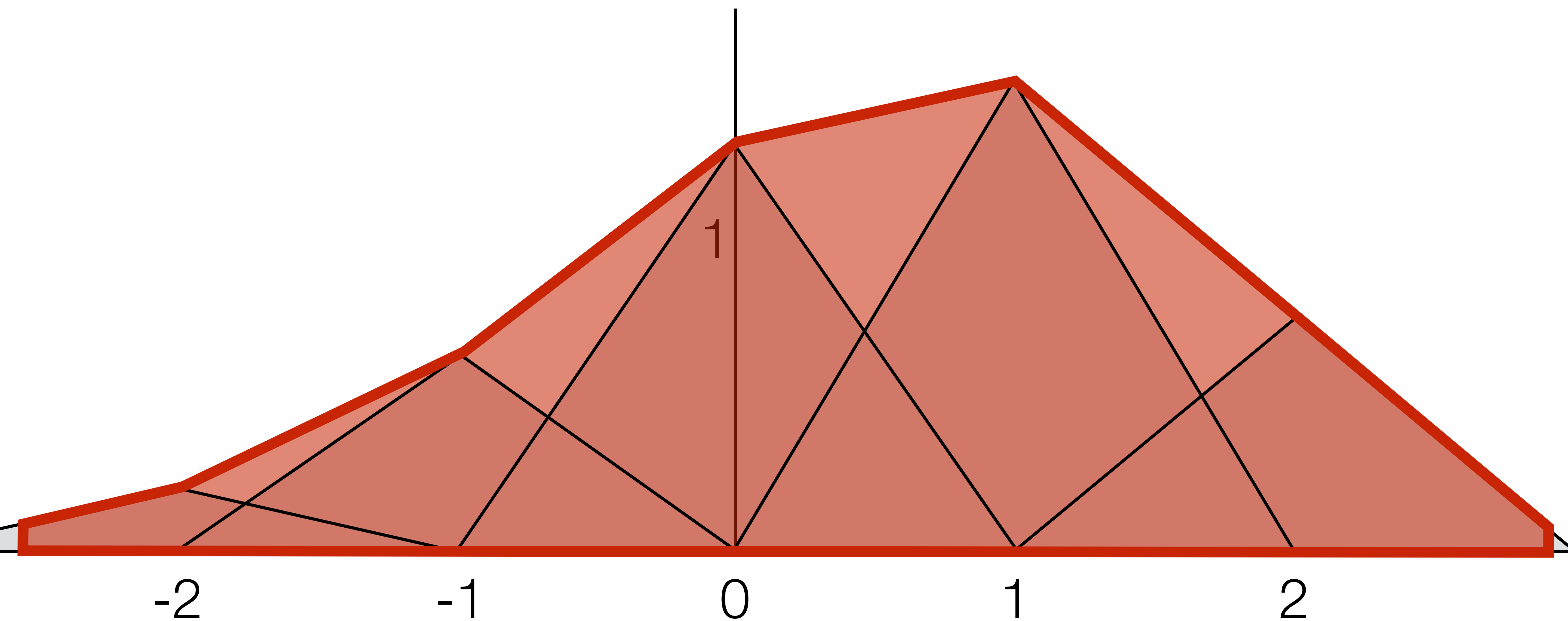


sums **shifts** **scales**

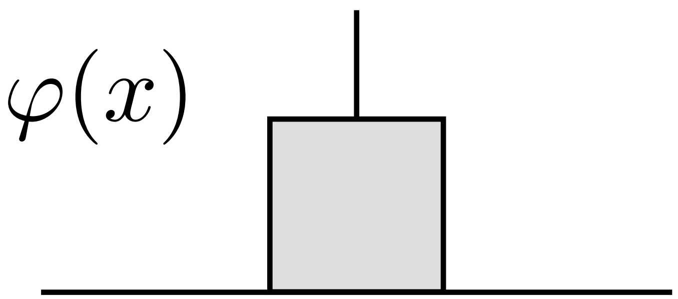
$$f(x) = \sum_i \varphi(x - i) c_i$$

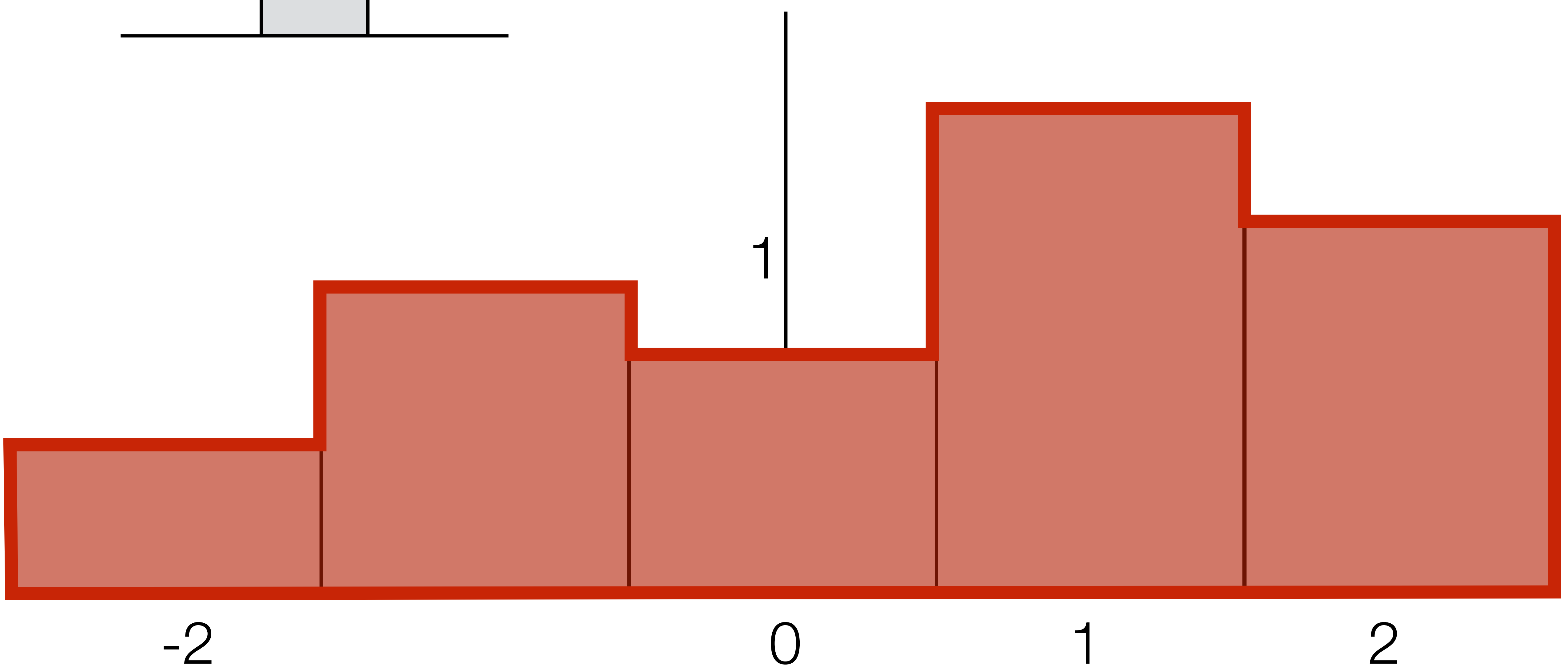
simple functions



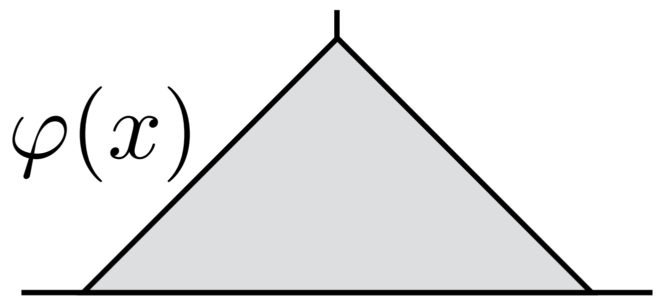


Example: nearest-neighbor interpolation



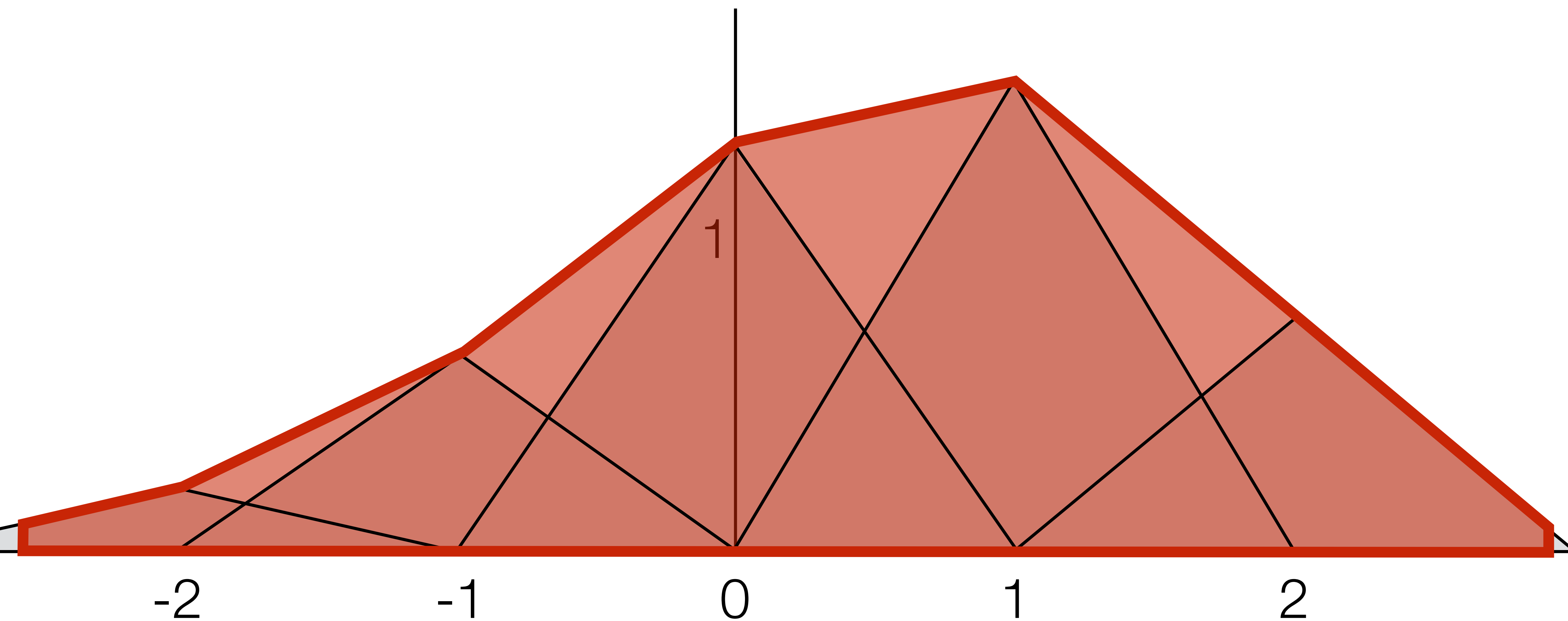


Example: linear interpolation

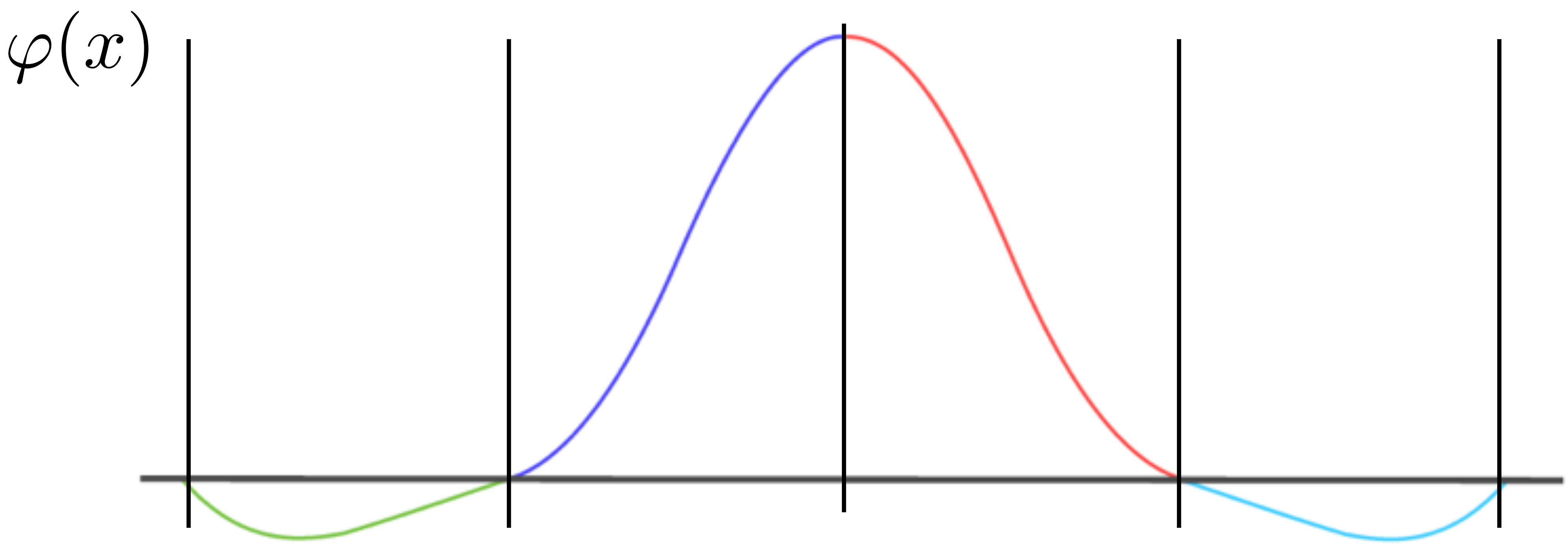


Alternative formulation:

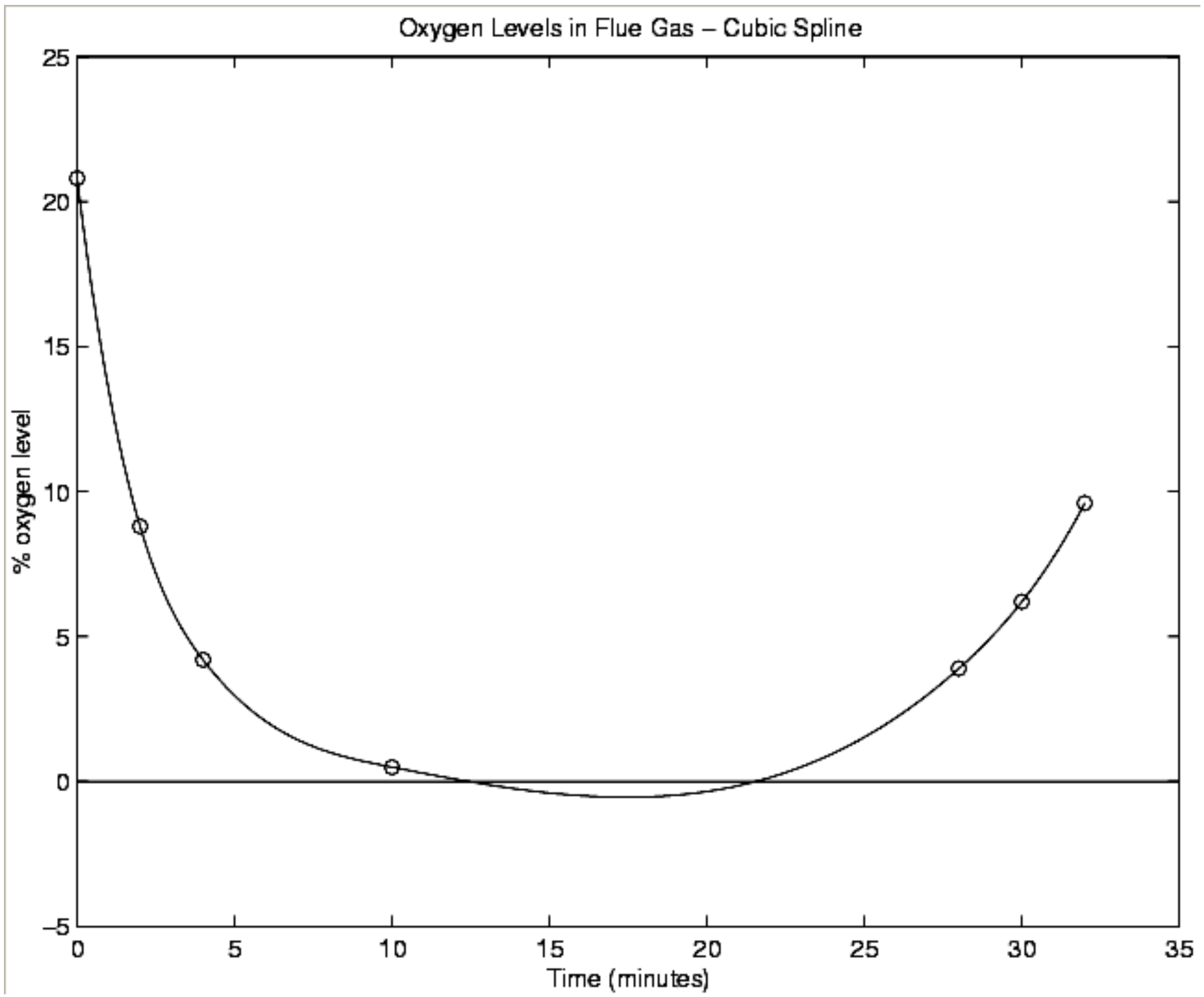
$f(x) = v_0(1 - x) + v_1 x$



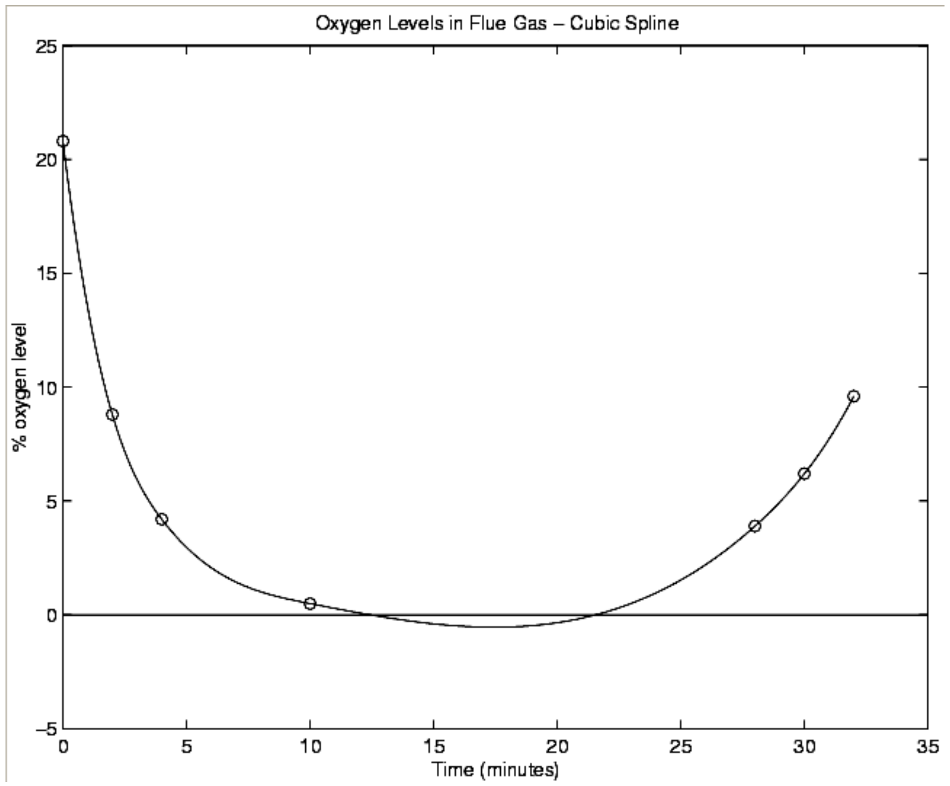
Cubic Interpolation

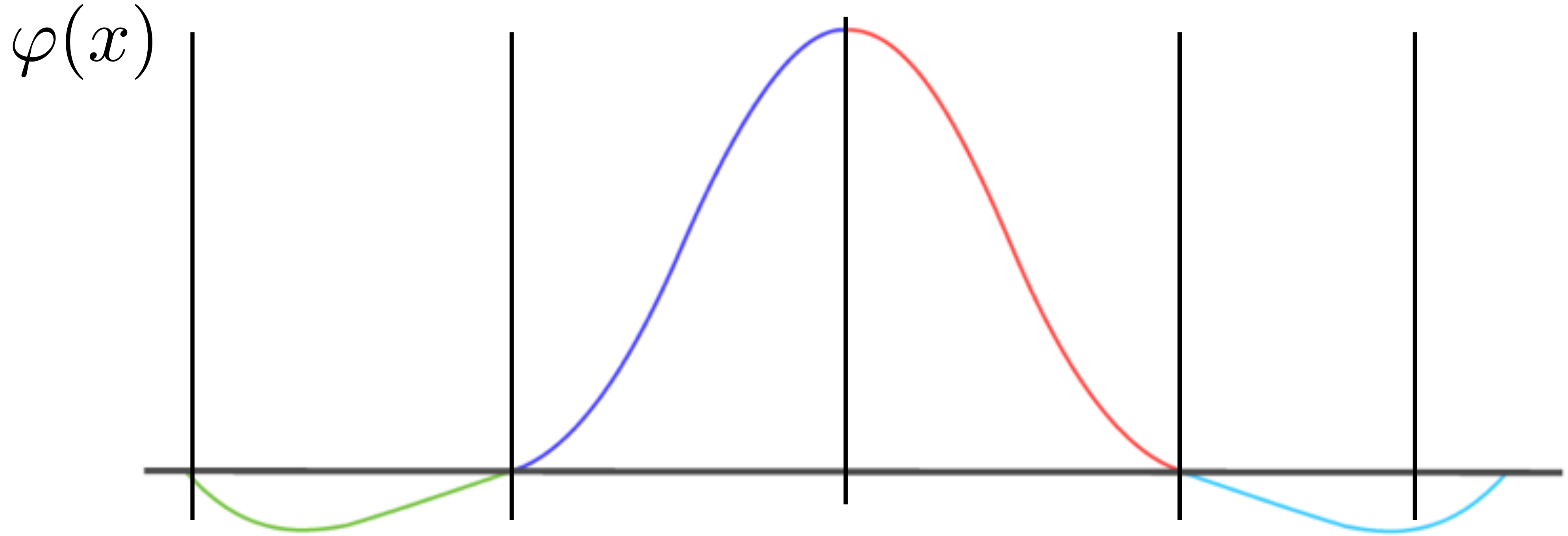


What is "Correct" Interpolation?

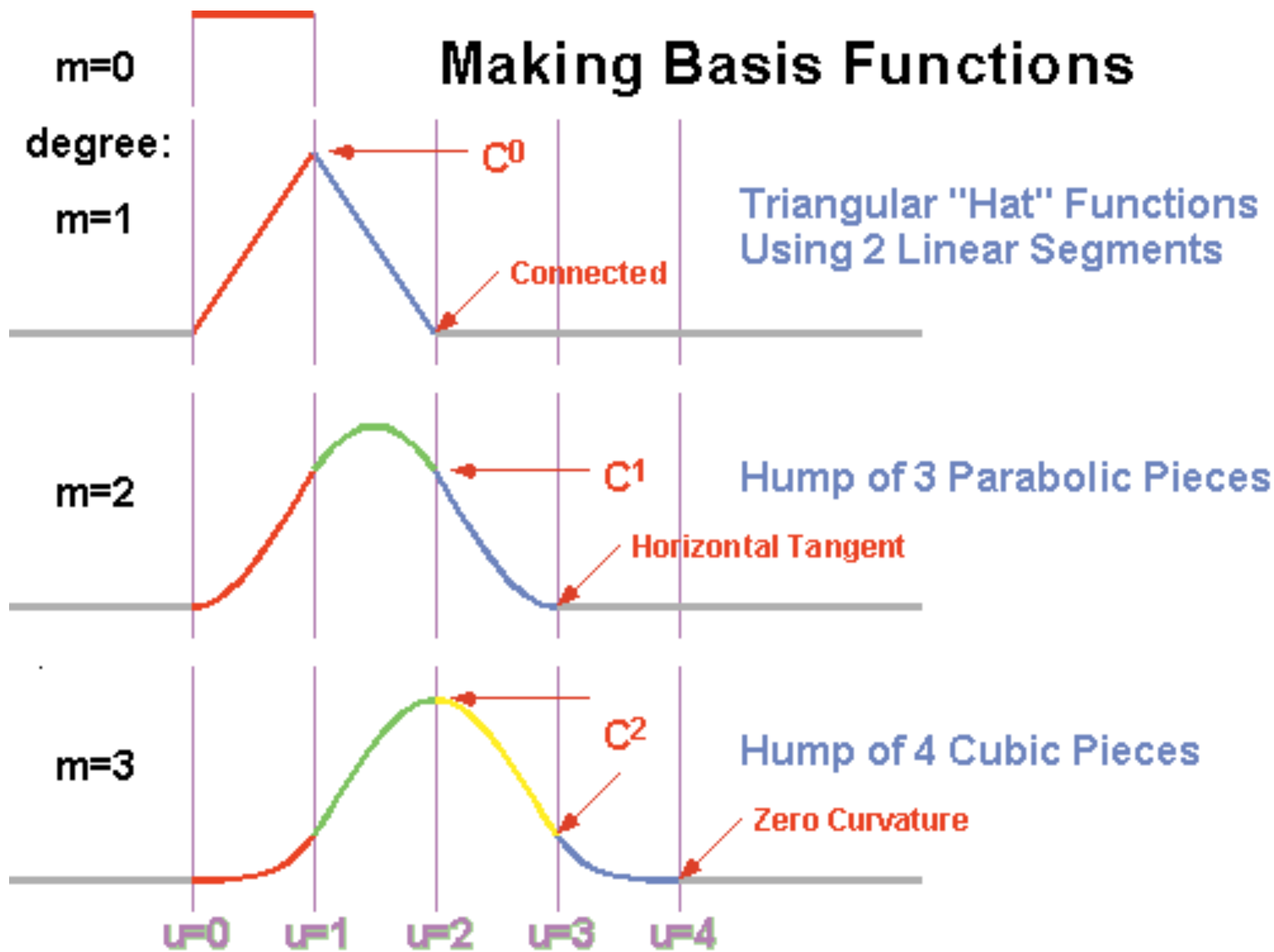


What is "Correct" Interpolation?





Cubic, (etc) **Approximation**



http://www.cs.berkeley.edu/~sequin/CS284/IMGS/makingbasisfunctions.gif

Why go through this trouble?

- Why not just define these functions “procedurally”?
  - At the end of the day they’re just arrays and if statements, after all

- **Because we can do math on those sums more easily**

Derivatives of finite-dimensional function spaces

$$f(x) = \sum_i c_i \varphi(x - i)$$

$$\frac{df}{dx}(x) = \frac{d}{dx} \sum_i c_i \varphi(x - i)$$

Derivatives of finite-dimensional function spaces

$$f(x) = \sum_i c_i \varphi(x - i)$$

$$\frac{df}{dx}(x) = \sum_i \frac{d}{dx} c_i \varphi(x - i)$$

Derivatives of finite-dimensional function spaces

$$f(x) = \sum_i c_i \varphi(x - i)$$

$$\frac{df}{dx}(x) = \sum_i c_i \frac{d\varphi}{dx}(x - i)$$

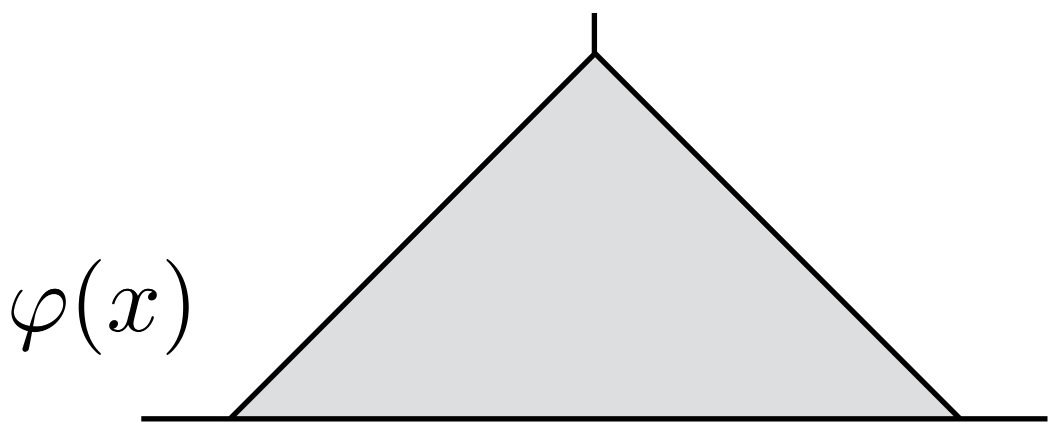
Derivatives of finite-dimensional function spaces

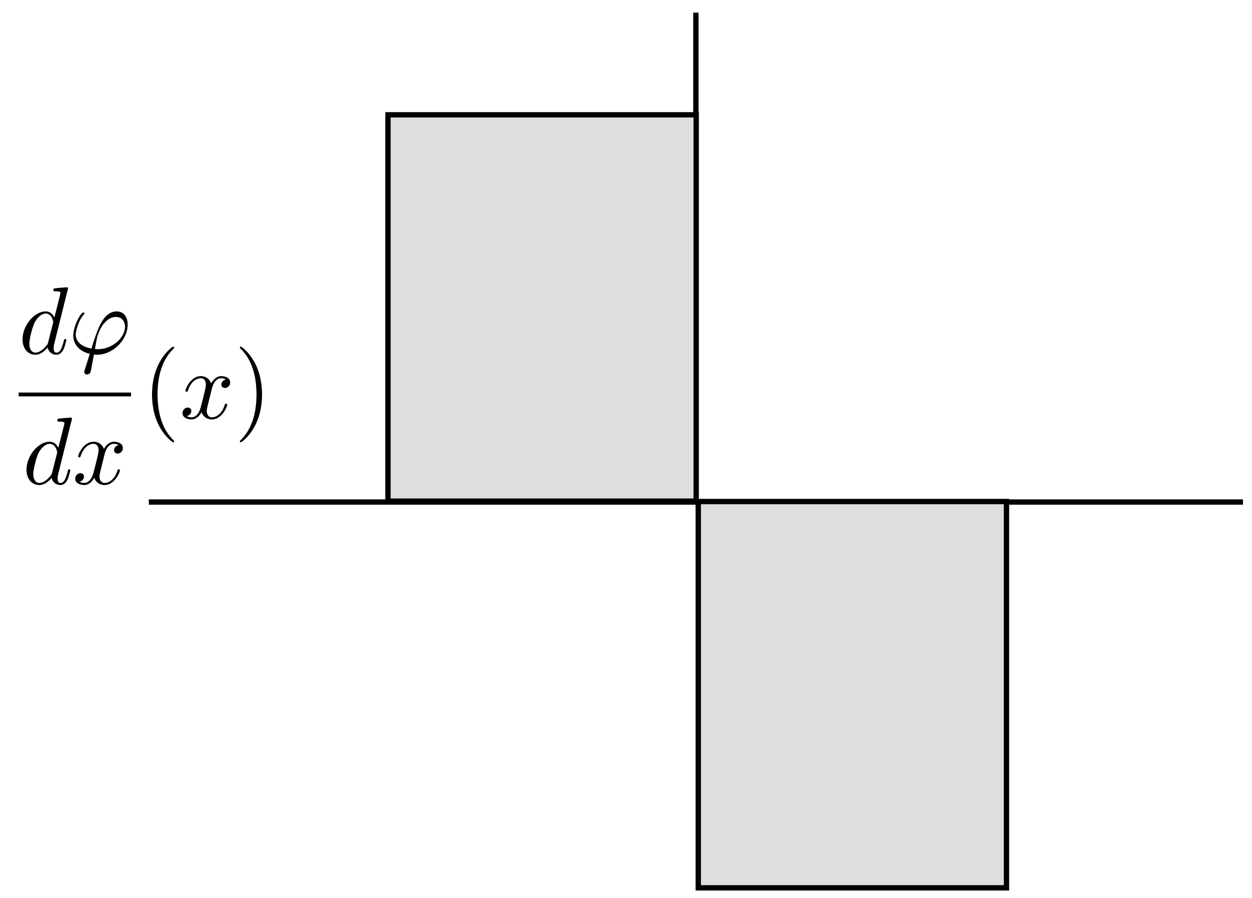
$$f(x) = \sum_i c_i \varphi(x - i)$$

$$\frac{df}{dx}(x) = \sum_i c_i \frac{d\varphi}{dx}(x - i)$$

- **Derivatives are just another type of function space where all we do is change the “simple function”**

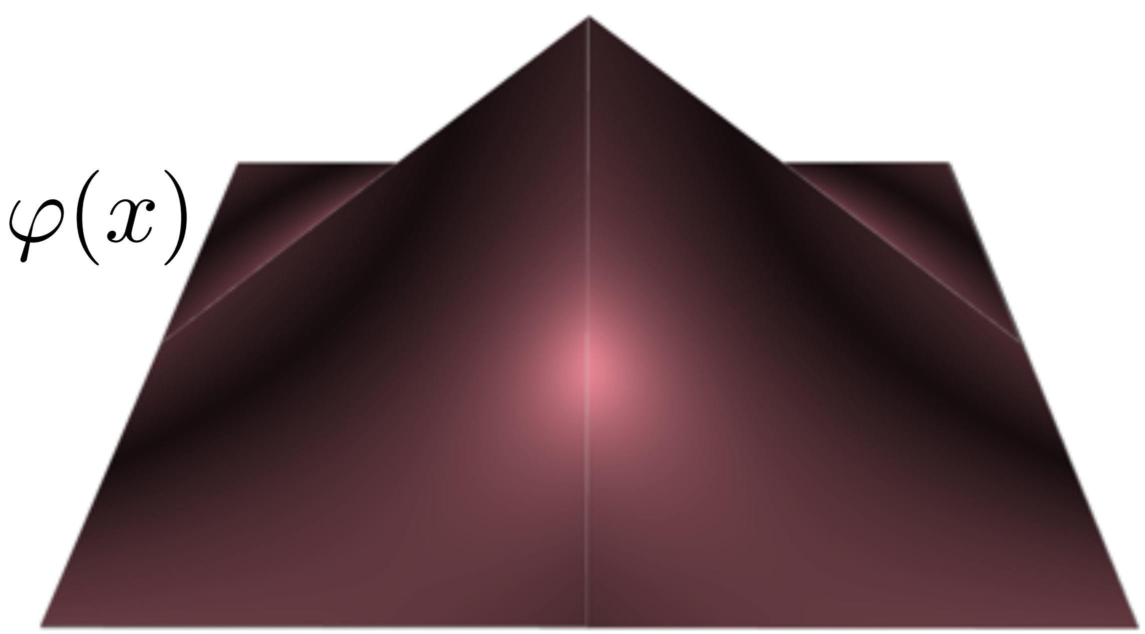
Derivatives of finite-dimensional function spaces





Multidimensional functions

$$f(x,y) = \begin{array}{cccc} v_{00} & (1-x) & (1-y) & + \\ v_{10} & (x) & (1-y) & + \\ v_{01} & (1-x) & (y) & + \\ v_{11} & (x) & (y) & \end{array}$$



Basis function
for bilinear interpolation

Interlude: The Gradient of a Function

$$\nabla f(\vec{x}) = \left[\begin{array}{c} \partial f / \partial x \\ \partial f / \partial y \end{array} \right]$$

But what is that?

Interlude: The Gradient of a Function

First we remember our friend the Taylor series:

$$f\left(\begin{bmatrix} x \\ y \end{bmatrix}\right) = f\left(\begin{bmatrix} x_0 \\ y_0 \end{bmatrix}\right) + \nabla f\left(\begin{bmatrix} x_0 \\ y_0 \end{bmatrix}\right)^T \begin{bmatrix} x - x_0 \\ y - y_0 \end{bmatrix} + \varepsilon$$

Now we ask ourselves:
if we move a little away from (x_0, y_0),
in what direction does f grow the fastest?

Interlude: The Gradient of a Function

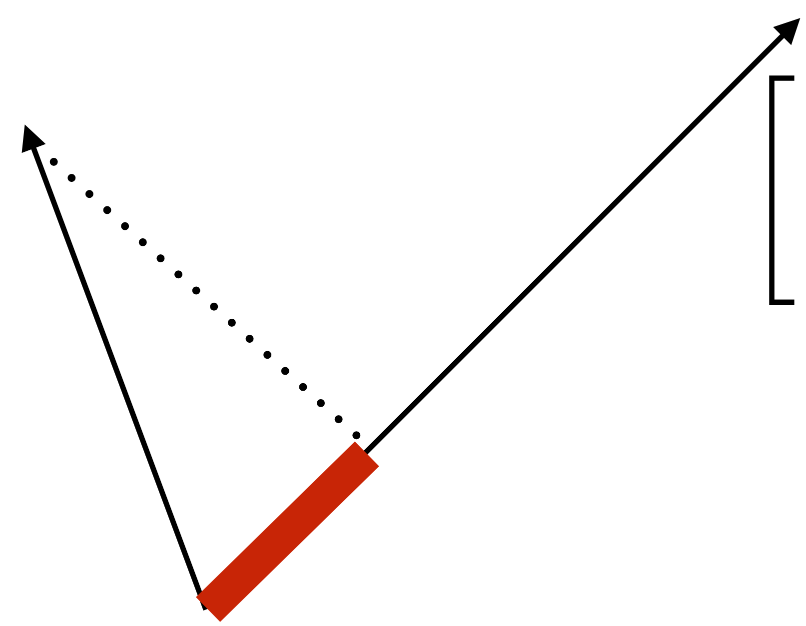
$$f\left(\left[\begin{array}{c} x \\ y \end{array}\right]\right) = f\left(\left[\begin{array}{c} x_0 \\ y_0 \end{array}\right]\right) + \nabla f\left(\left[\begin{array}{c} x_0 \\ y_0 \end{array}\right]\right)^T \left[\begin{array}{c} x - x_0 \\ y - y_0 \end{array}\right] + \varepsilon$$

$$\nabla f\left(\left[\begin{array}{c} x_0 \\ y_0 \end{array}\right]\right)^T \left[\begin{array}{c} dx \\ dy \end{array}\right]$$

$$= \left[\begin{array}{c} \partial f / \partial x \\ \partial f / \partial y \end{array}\right]^T \left[\begin{array}{c} dx \\ dy \end{array}\right]$$

Interlude: The Gradient of a Function

$$\max \begin{bmatrix} \partial f / \partial x \\ \partial f / \partial y \end{bmatrix}^T \begin{bmatrix} dx \\ dy \end{bmatrix}$$

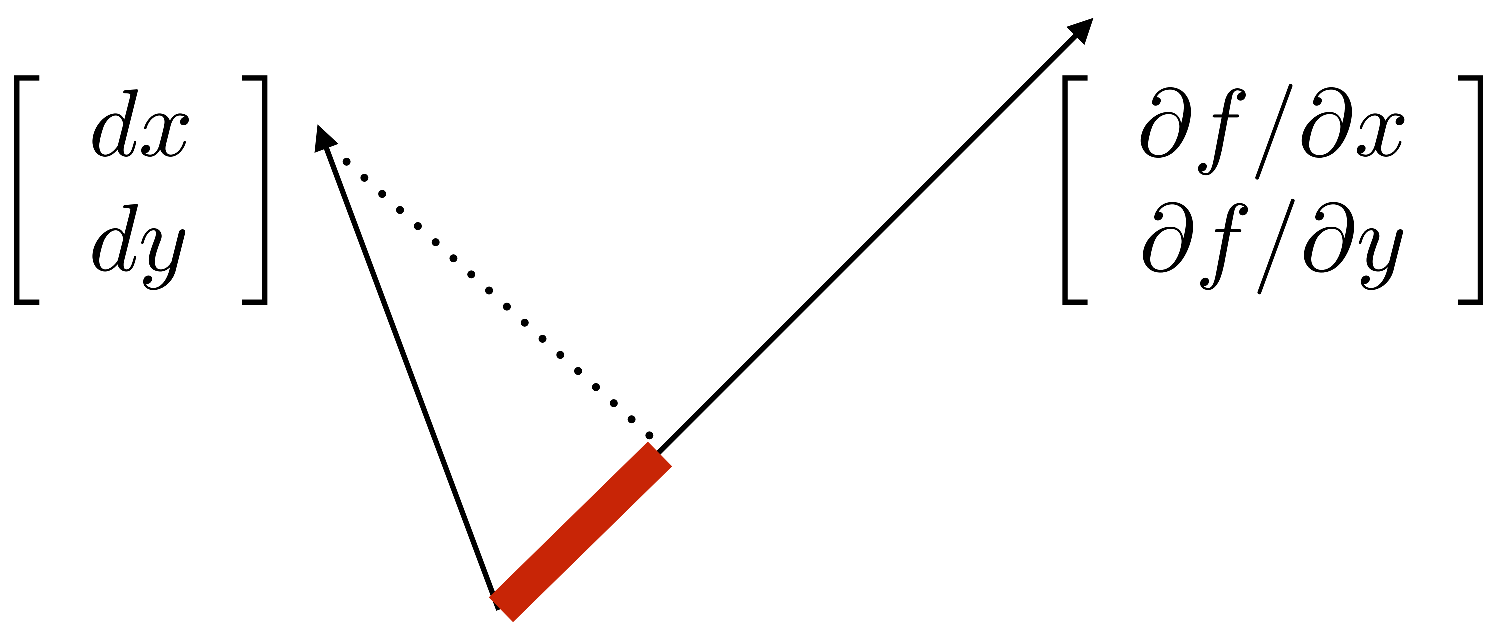


$$\begin{bmatrix} dx \\ dy \end{bmatrix}$$

$$\begin{bmatrix} \partial f / \partial x \\ \partial f / \partial y \end{bmatrix}$$

Interlude: The Gradient of a Function

$$\max \begin{bmatrix} \partial f / \partial x \\ \partial f / \partial y \end{bmatrix}^T \begin{bmatrix} dx \\ dy \end{bmatrix} \qquad \begin{bmatrix} dx \\ dy \end{bmatrix} = \frac{\nabla f}{|\nabla f|}$$



Interlude: The Gradient of a Function

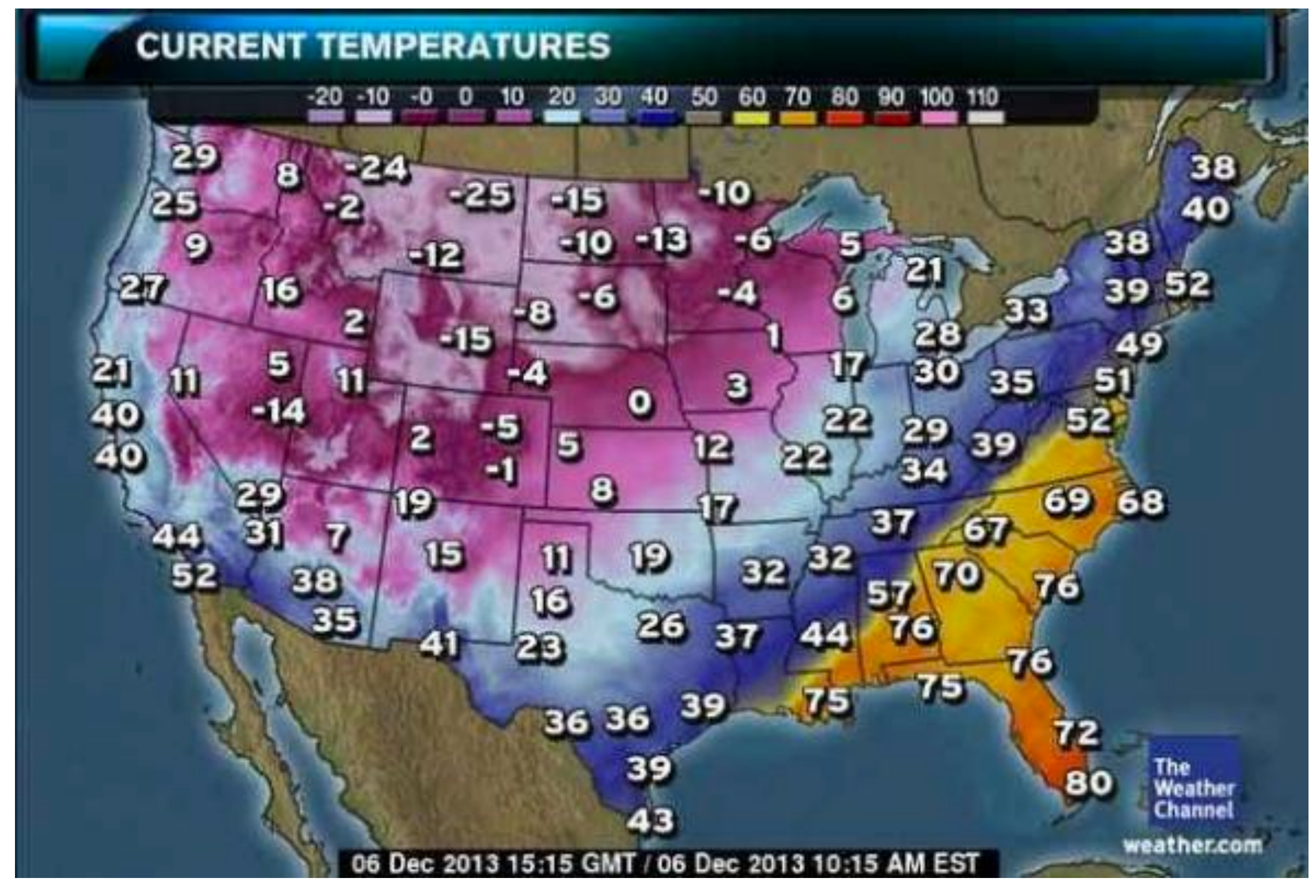
$$\max \begin{bmatrix} \partial f / \partial x \\ \partial f / \partial y \end{bmatrix}^T \begin{bmatrix} dx \\ dy \end{bmatrix} \qquad \begin{bmatrix} dx \\ dy \end{bmatrix} = \frac{\nabla f}{|\nabla f|}$$

The gradient **points in the direction of greatest increase** and its **length is the rate of greatest increase**

Visualizing Scalar Fields

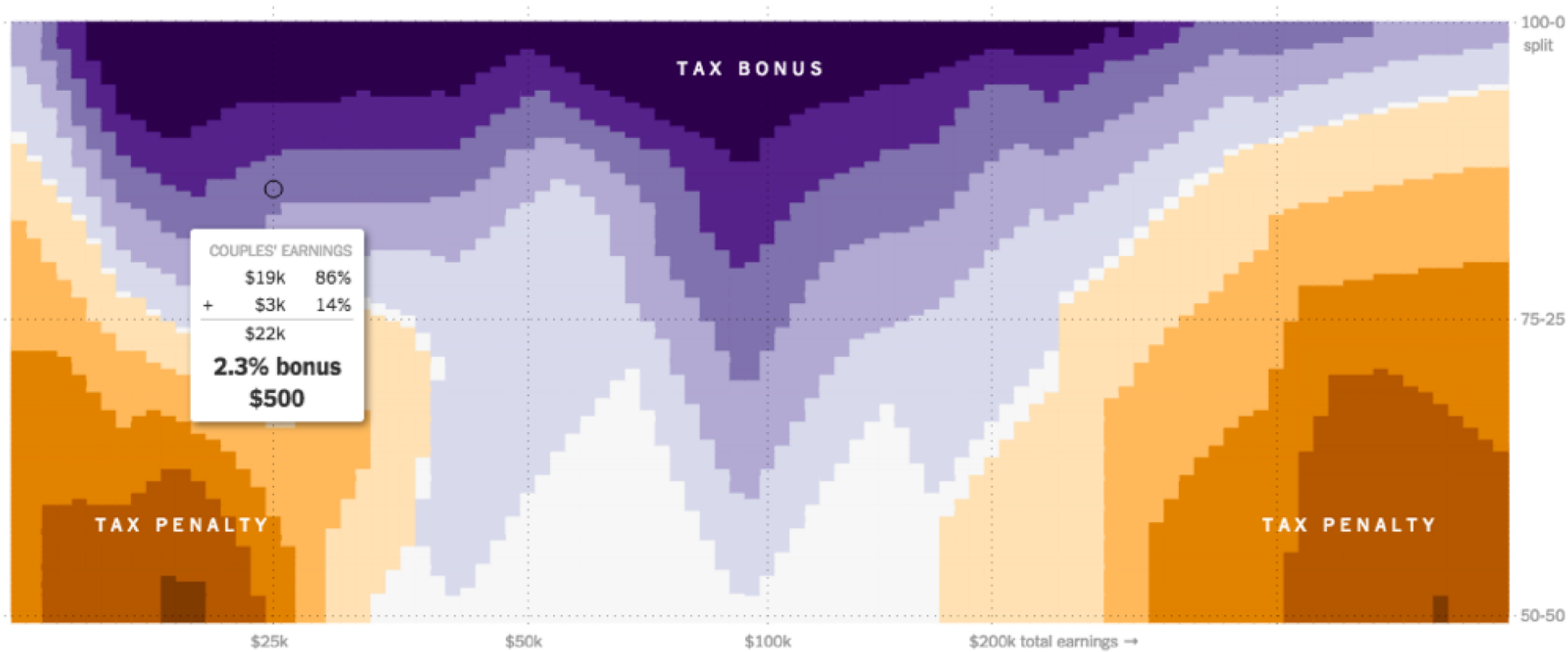
Colormapping

- “Default” strategy:
  - create color scale using the **range** of the function as the **domain** of the scale
  - create a position scale to convert **from the domain of the function to positions on the screen**
  - set the pixel color according to the scale



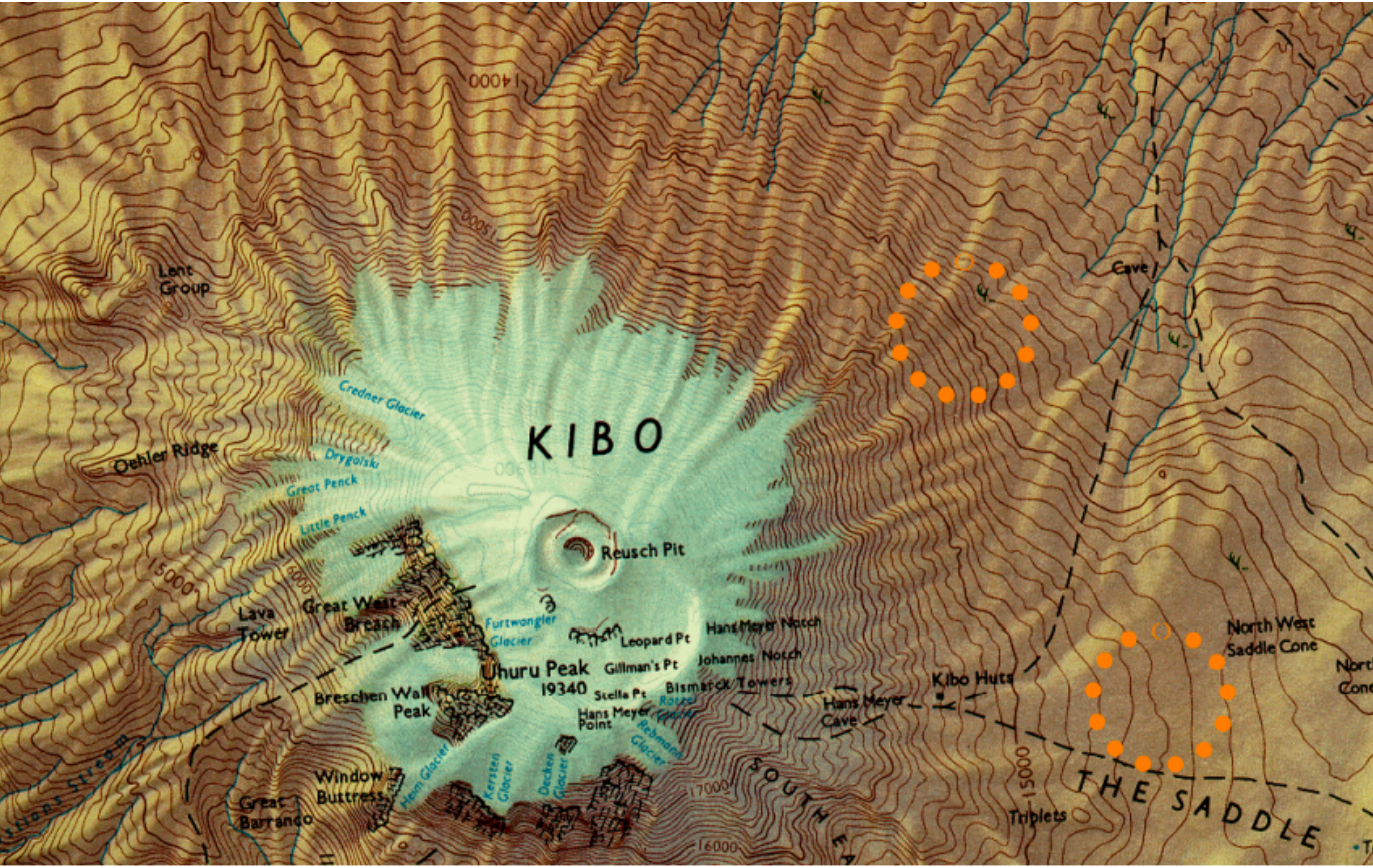
Colormapping guidelines apply!

Applies to "abstract" spaces too

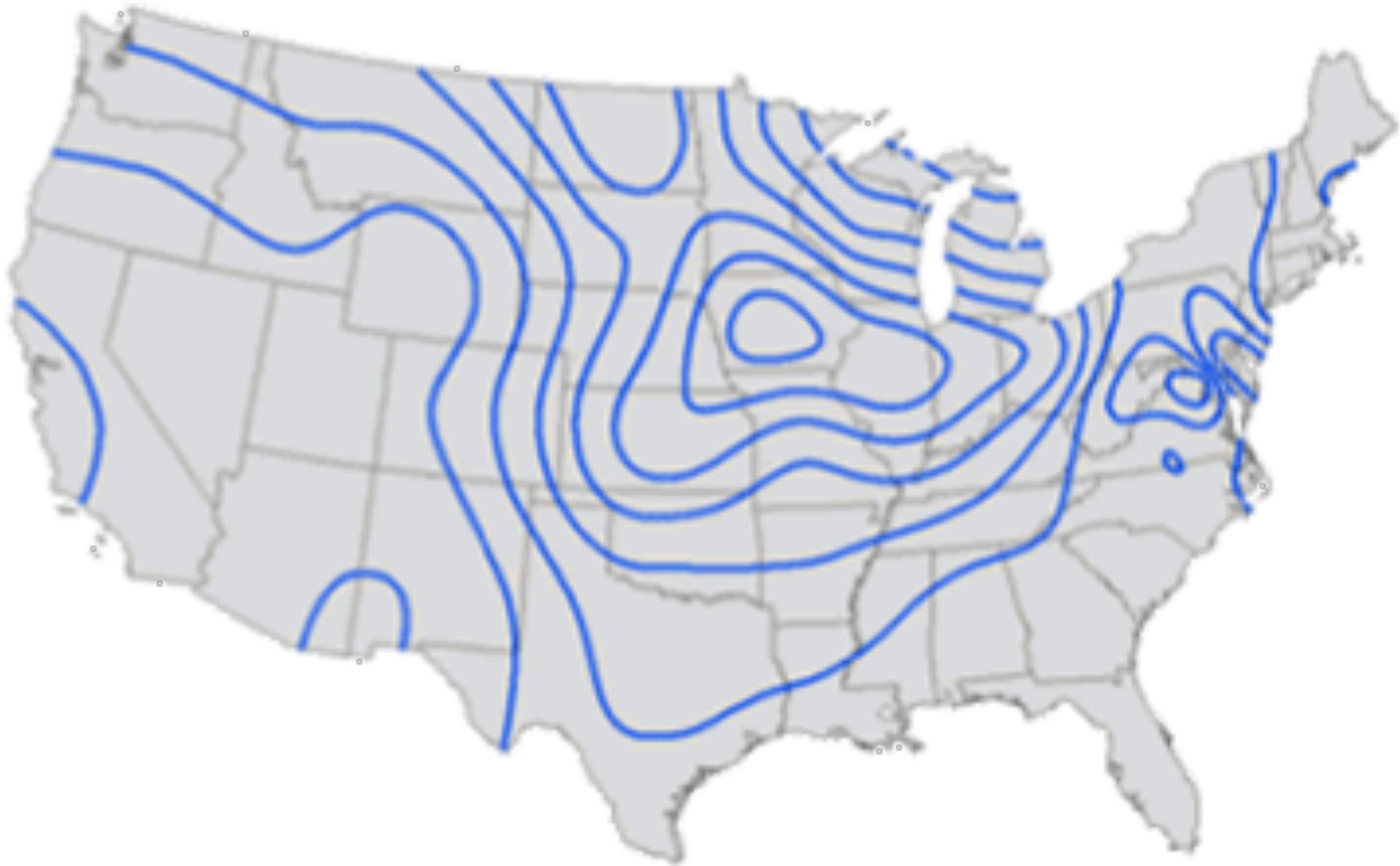


http://www.nytimes.com/interactive/2015/04/16/upshot/marriage-penalty-couples-income.html?abt=0002&abg=0

Contouring: isolines



Contouring: isolines



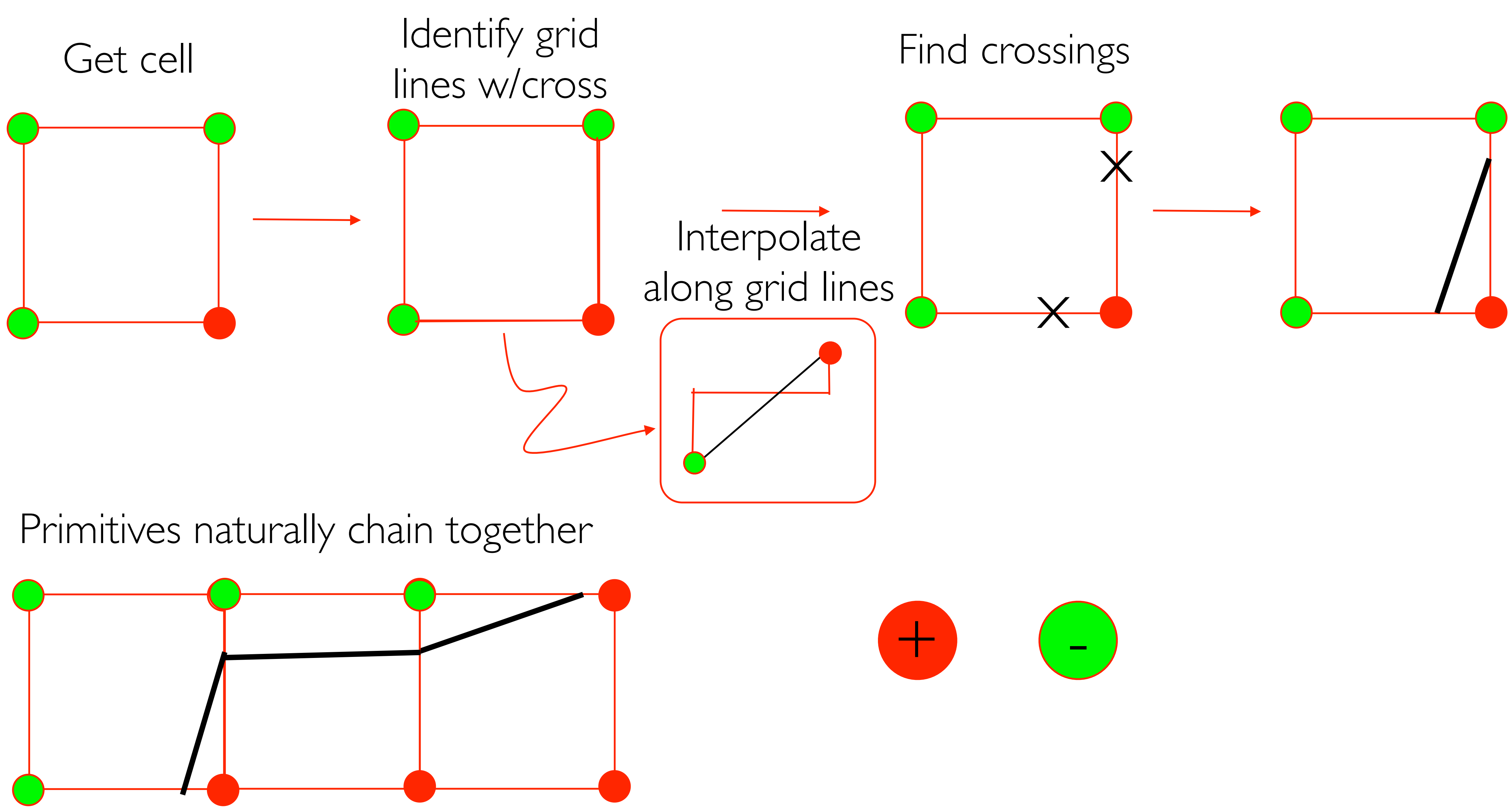
http://ryanhill1.blogspot.com/2011/07/isoline-map.html

Contouring: isolines

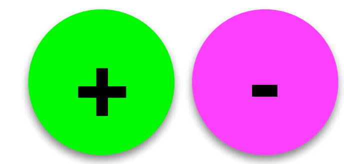
How do we compute them?

Approach to Contouring in 2D

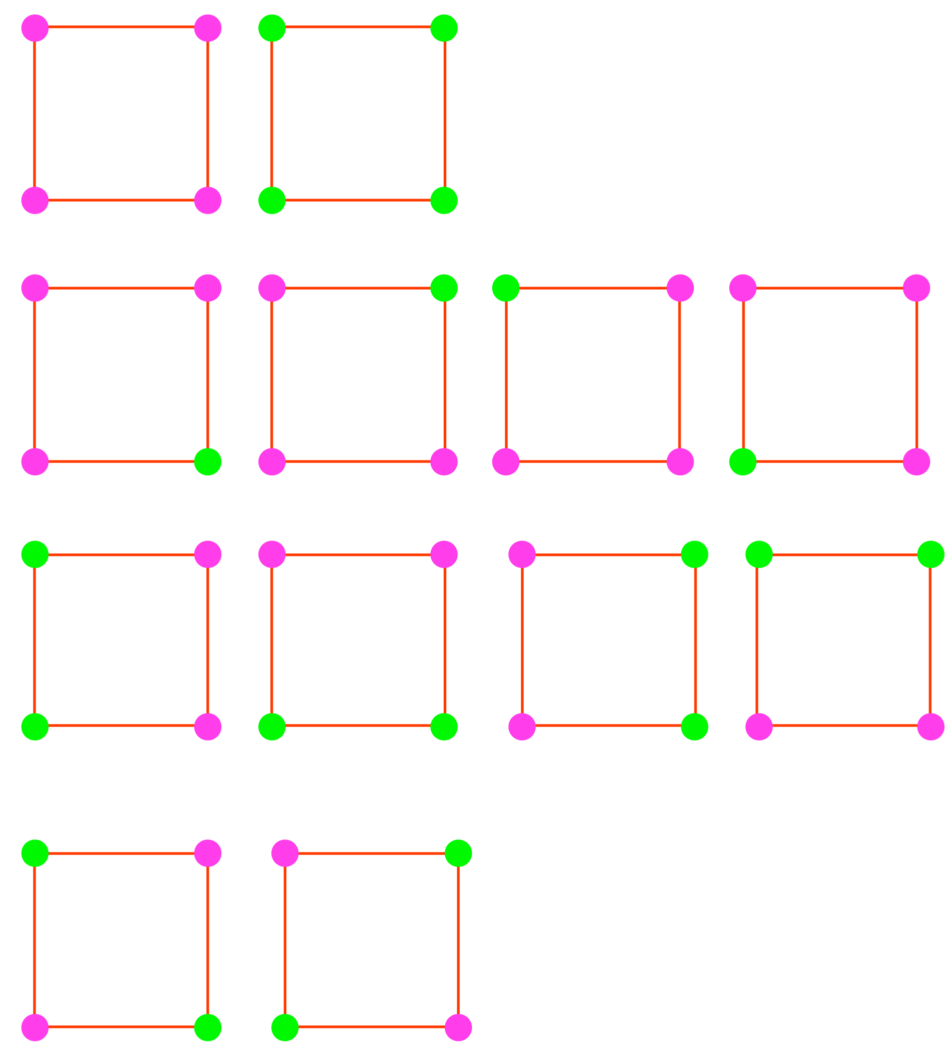
- Contour must cross every grid line connecting two grid points of opposite sign



Cases



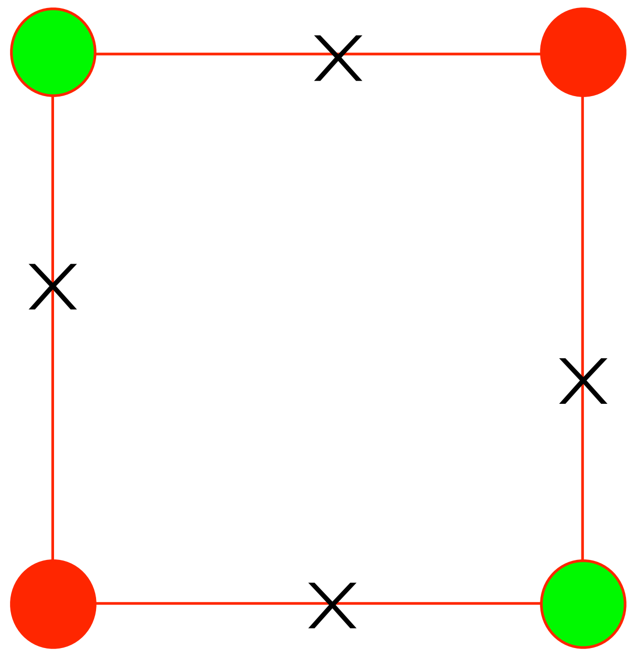
Case	Polarity	Rotation	Total
No Crossings	x2		2
Singlet	x2	x4	8
Double adjacent	x2	x2 (4)	4
Double Opposite	x2	x1 (2)	2
			$16 = 2^4$



(x2 for polarity)

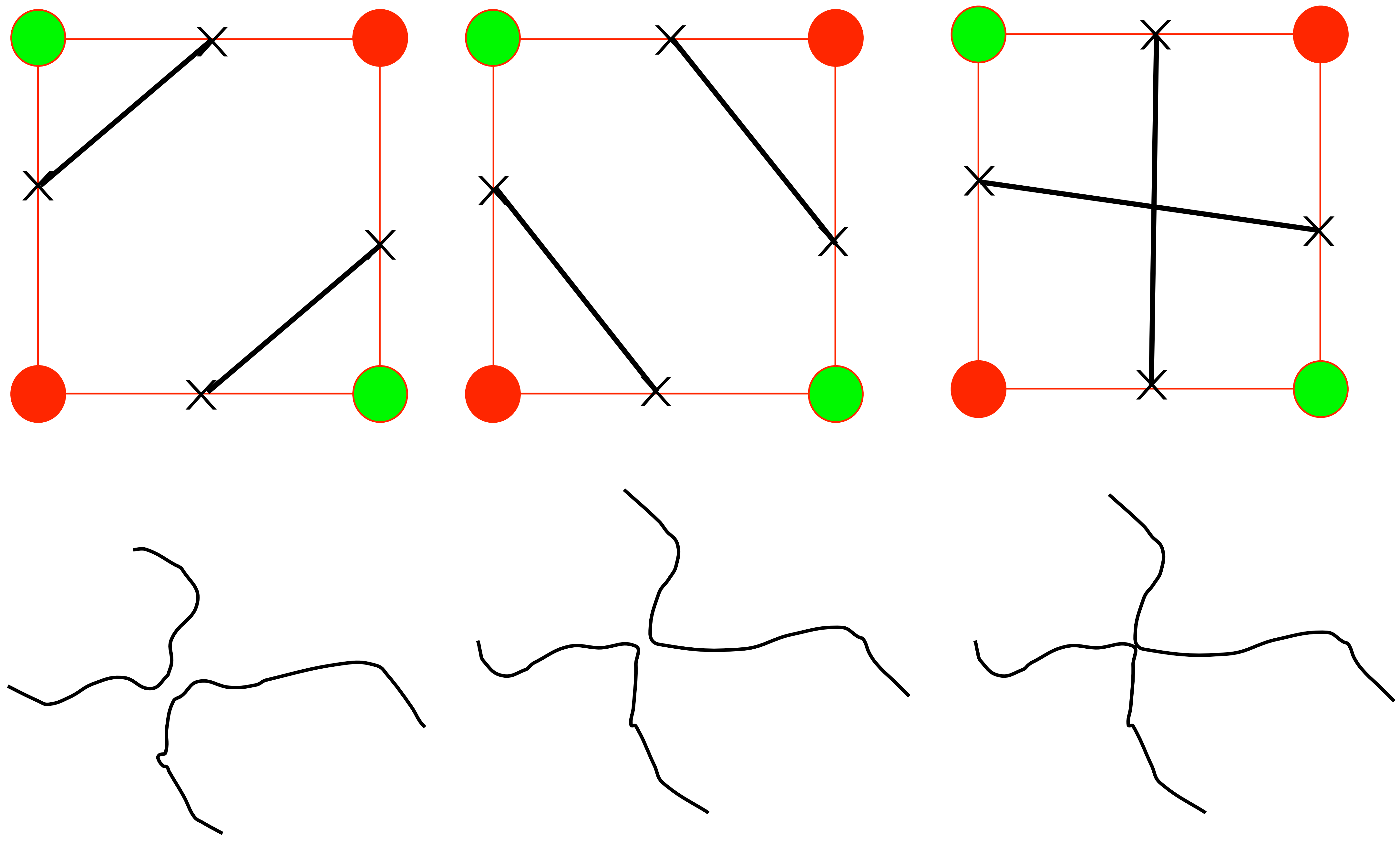
Ambiguities

- How to form lines?



Ambiguities

- Right or Wrong?



Contouring: isolines