Data Visualization

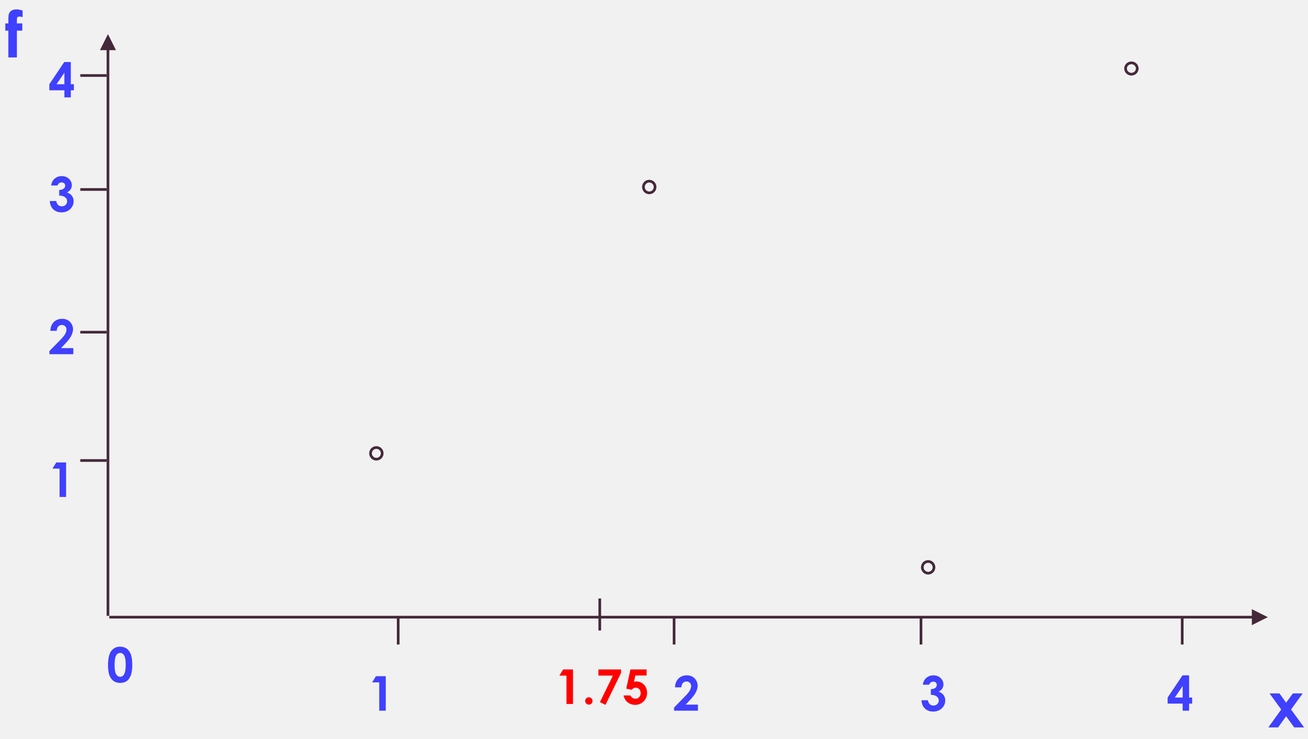
Visualization Techniques -

1D Scalar Data

2D Scalar Data

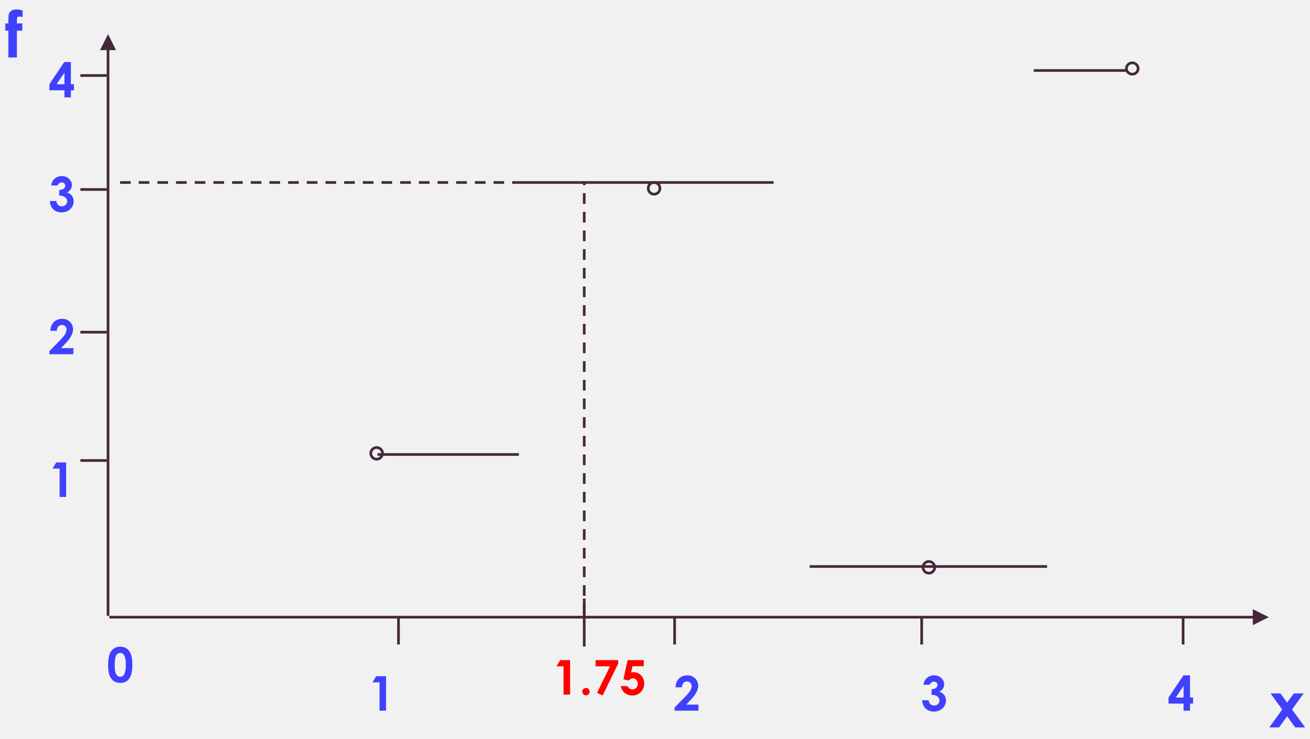
Visualization Techniques - One Dimensional Scalar Data

1D Interpolation -The Problem



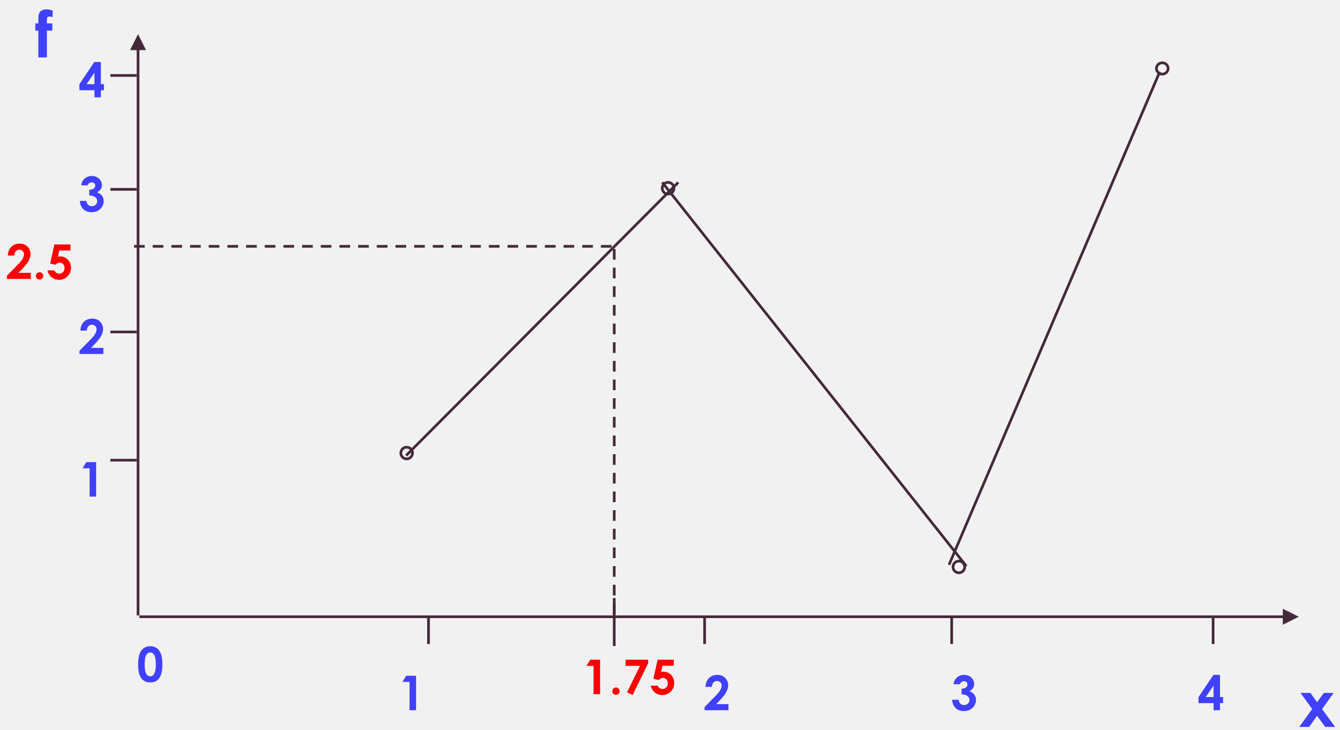
Given (x_1,f_1), (x_2,f_2), (x_3,f_3), (x_4,f_4) - estimate the value of f at other values of x - say, x*. Suppose x*=1.75

Nearest Neighbour



Take f-value at x* as f-value of nearest data sample.
So if x* = 1.75, then f estimated as 3

Linear Interpolation



Join data points with straight lines- read off f-value corresponding to x*.. in the case that x*=1.75, then f estimated as 2.5

Linear Interpolation - Doing the Calculation

Suppose x^* lies between x_1 and x_2. Then apply the transformation:

$$t = (x^*-x_1)/(x_2-x_1)$$

so that t goes from 0 to 1.

$t=(1.75-1)/(2-1)=0.75$

$$f(x^*) = (1-t)\, f_1 + t\, f_2$$

$f(1.75)=0.25*1+0.75*3$
$=2.5$

The functions $j(t)=1-t$ and $k(t)=t$ are *basis functions*.

OR, saving a multiplication:

$$f(x^*) = f_1 + t\,(f_2 - f_1)$$

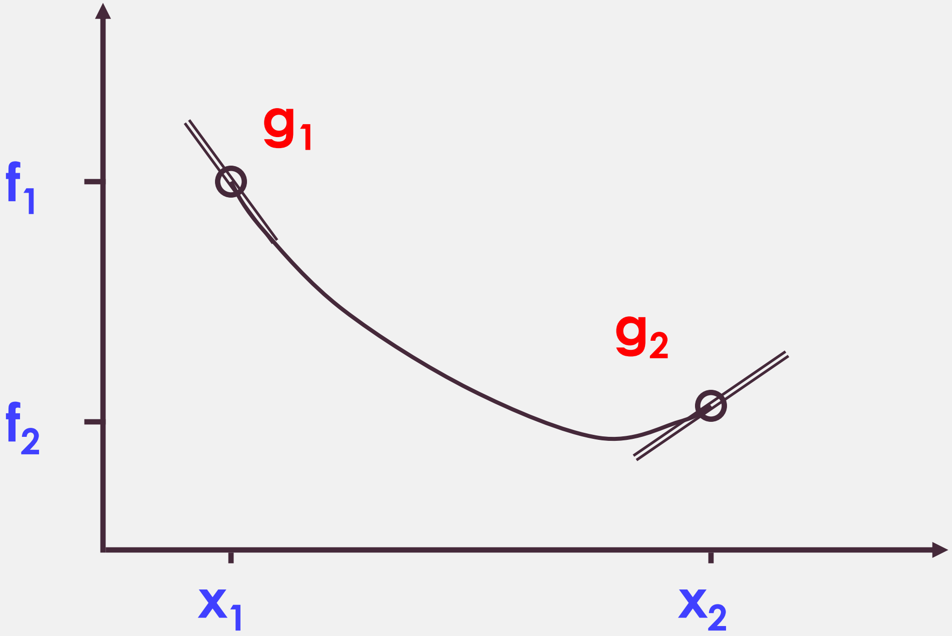
$f(1.75)=1+0.75*(3-1)$
$=2.5$

Nearest Neighbour and Linear Interpolation

- Nearest Neighbour
  - Very fast : no arithmetic involved
  - Continuity : discontinuous value
  - Bounds : bounds fixed at data extremes
- Linear Interpolation
  - Fast : one multiply, one divide
  - Continuity : value only continuous, not slope (C^0)
  - Bounds : bounds fixed at data extremes

Drawing a Smooth Curve

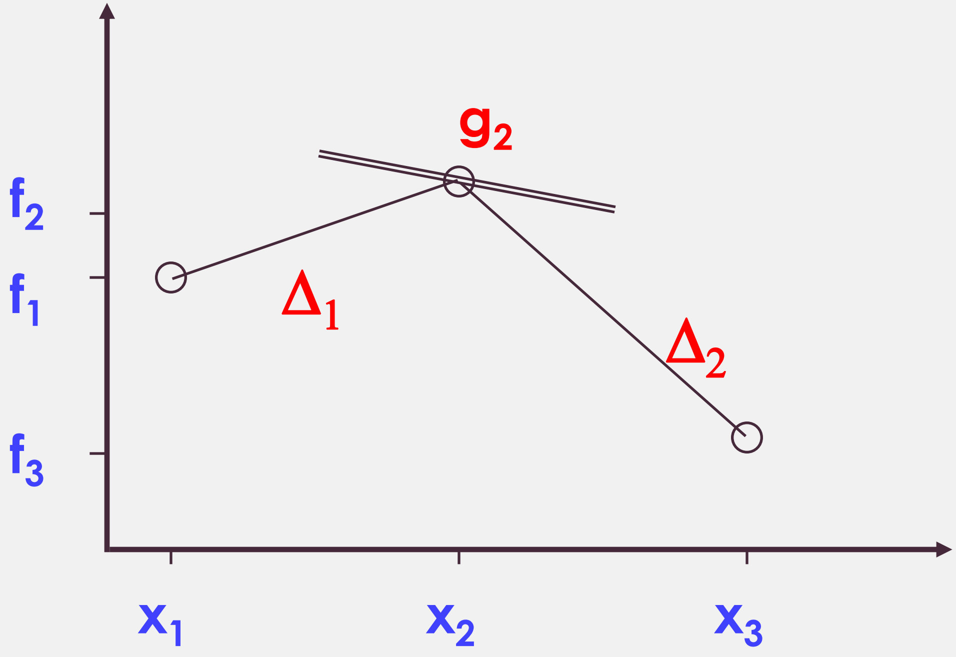
- **Rather than straight line between points, we create a curve piece**



We estimate the slopes g_1 and g_2 at the data points, and construct curve which has these values and these slopes at end-points

Slope Estimation

- Slopes estimated as some average of the slopes of adjacent chords - eg to estimate slope at x_2

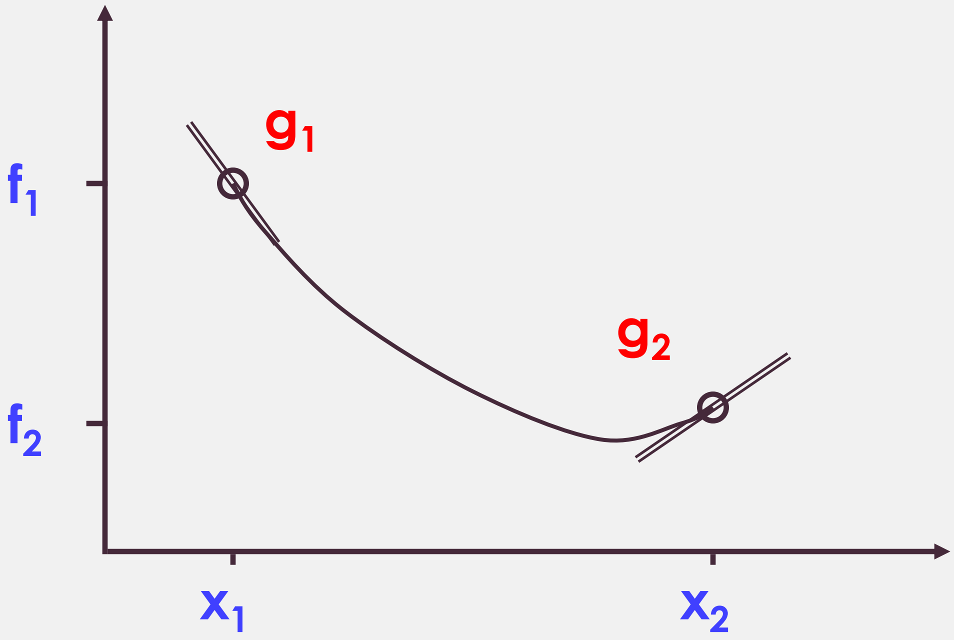


g_2 usually arithmetic mean (ie average) of Δ_1, Δ_2

$$\Delta_1 = (f_2 - f_1)/(x_2 - x_1)$$

Piecewise Cubic Interpolation

Once the slopes at x_1 and x_2 are known, this is sufficient to define a unique cubic polynomial in the interval $[x_1,x_2]$



$$
\begin{aligned}
f(x) &= c_1(x) * f_1 \\
&+ c_2(x) * f_2 \\
&+ h*(d_1(x) * g_1 \\
&- d_2(x) * g_2)
\end{aligned}
$$

$c_i(x)$, $d_i(x)$ are cubic Hermite basis functions, $h = x_2 - x_1$.

Cubic Hermite Basis Functions

Here they are:

Again set $t = (x - x_1)/(x_2 - x_1)$

$c_1(t) = 3(1-t)^2 - 2(1-t)^3$

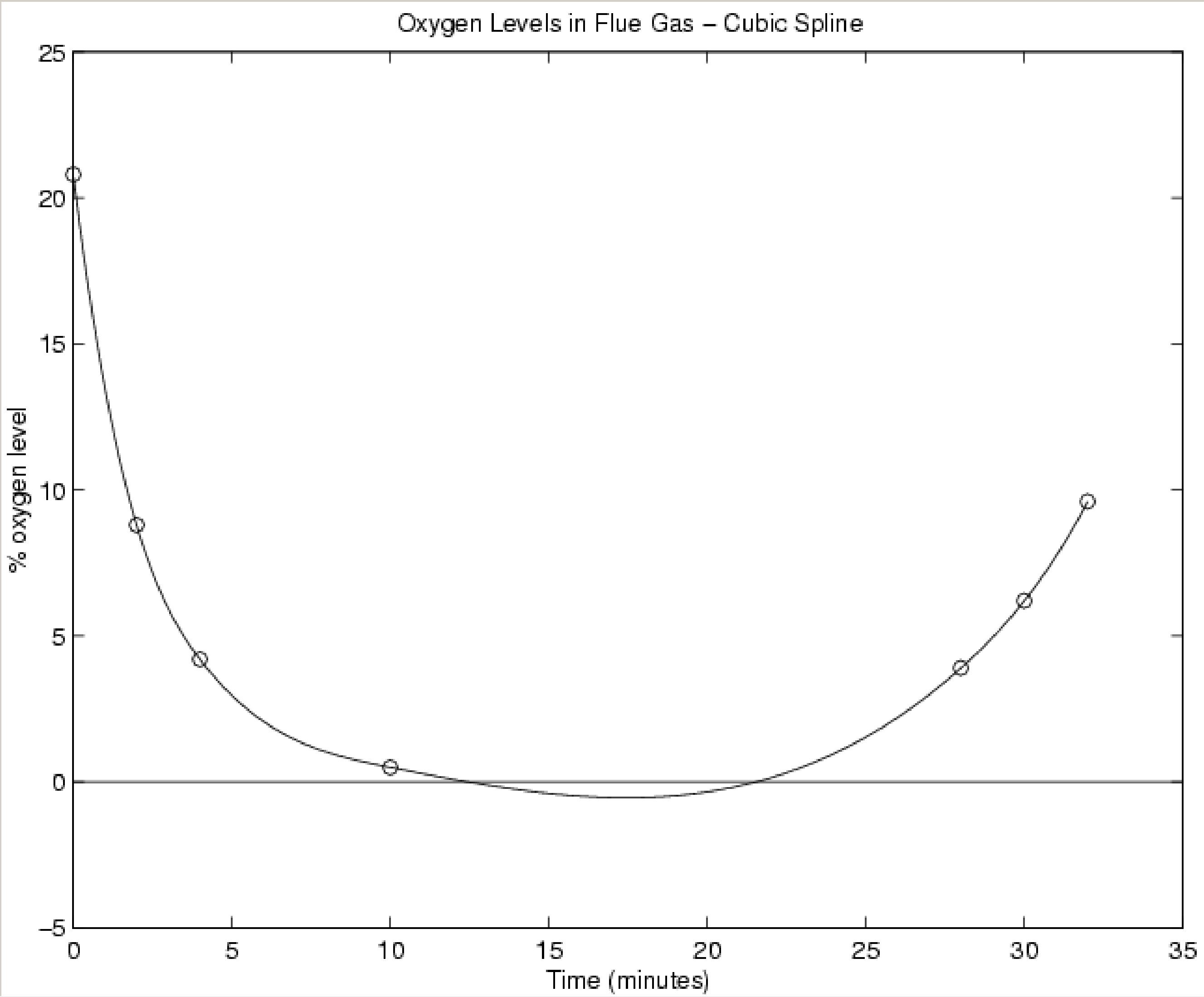
$c_2(t) = 3t^2 - 2t^3$

$d_1(t) = (1-t)^2 - (1-t)^3$

$d_2(t) = t^2 - t^3$

Check the values
at $x = x_1, x_2$ (ie $t=0,1$)

Coal data - cubic interpolation



Piecewise Cubic Interpolation

- More computation needed than with nearest neighbour or linear interpolation.
- Continuity: slope continuity (C^1) by construction - and cubic splines will give second derivative continuity (C^2)
- Bounds: bounds not controlled generally - eg if arithmetic mean used in slope estimation...

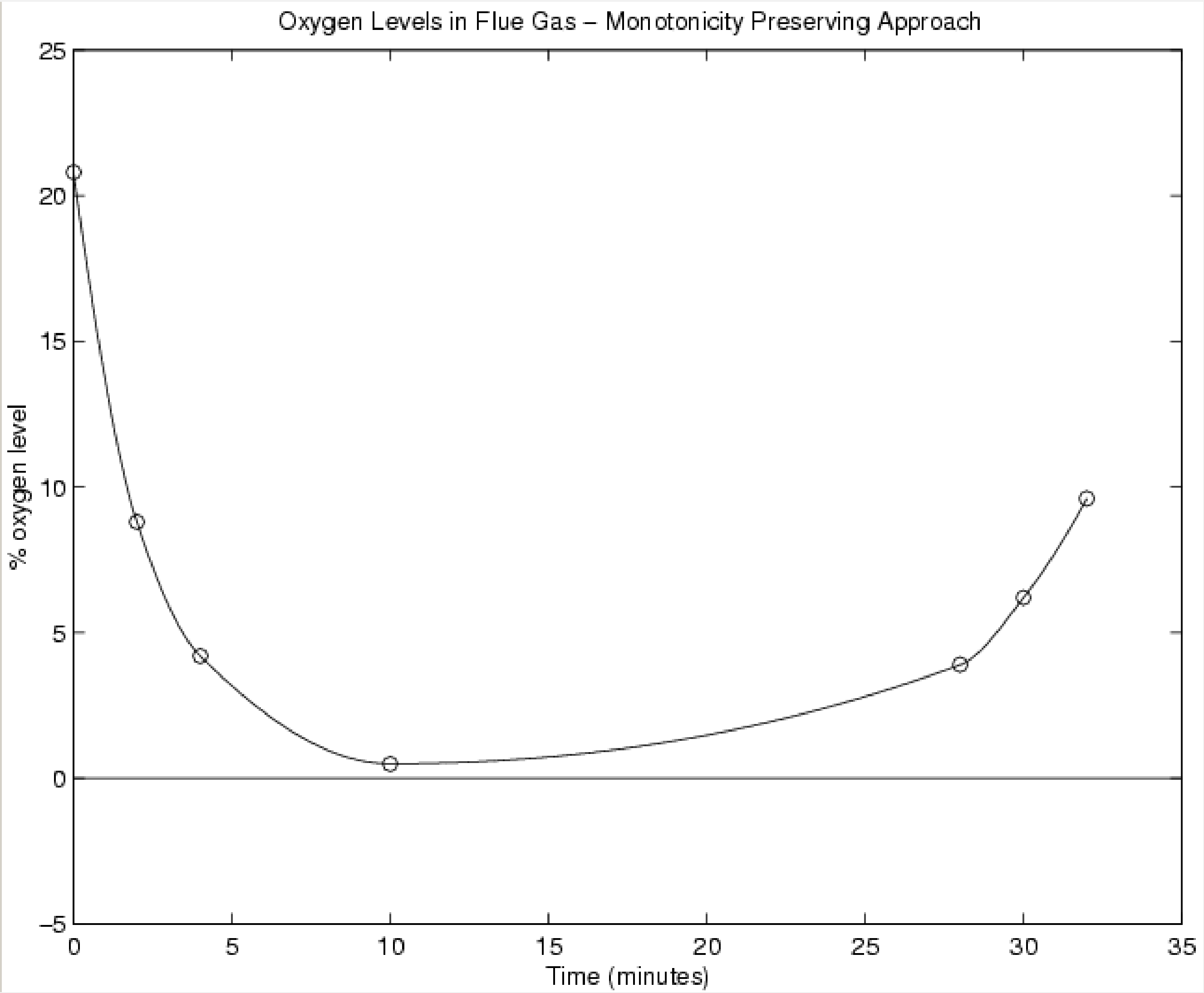
Shape Control

- However special choices for slope estimation do give control over shape
- If the harmonic mean is used

$$1/g_2 = 0.5 \, (1/\Delta_1 + 1/\Delta_2)$$

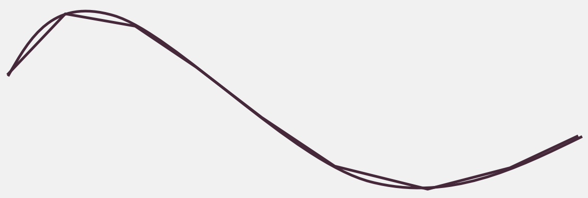
then we find that f(x) lies within the bounds of the data

Coal data – keeping within the bounds of the data



Rendering Line Graphs

- The final rendering step is straightforward
- We can assume that the underlying graphics system will be able to draw straight line segments
- Thus the linear interpolation case is trivial
- For curves, we do an approximation as sequence of small line segments



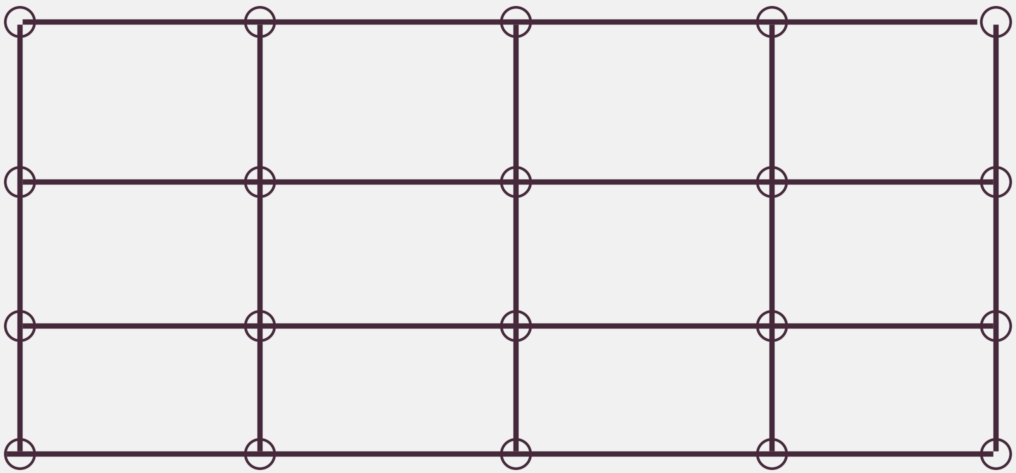
Background Study

- Take the coal furnace data from lecture 2
  - Investigate how to present this as a graph in Excel (not easy!)
  - After learning gnuplot in the practical class, create a graph of the coal data using gnuplot
- Look for Java applets on the Web that will create line graphs

Visualization Techniques
Two Dimensional Scalar Data

2D Interpolation - Rectangular Grid

- Suppose we are given data on rectangular grid:



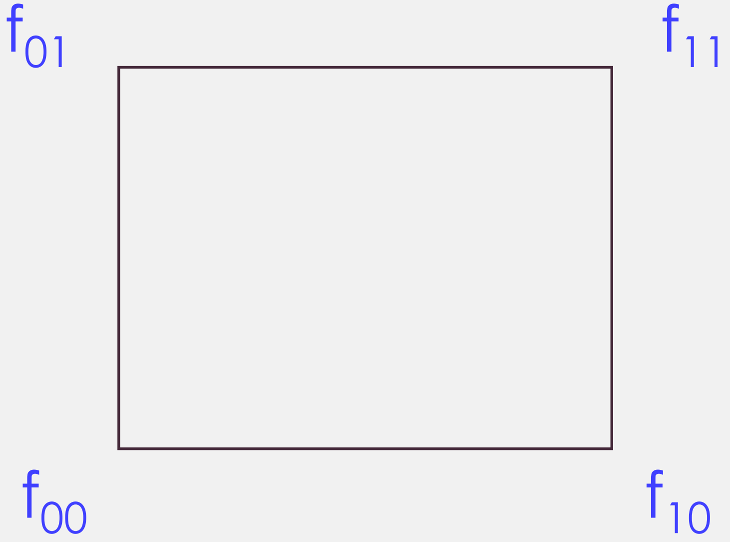
f given at each grid point;
data enrichment fills out the empty spaces by interpolating values within each cell

Nearest Neighbour Interpolation

- Straightforward extension from 1D: take f-value from nearest data sample
- No continuity
- Bounds fixed at data extremes

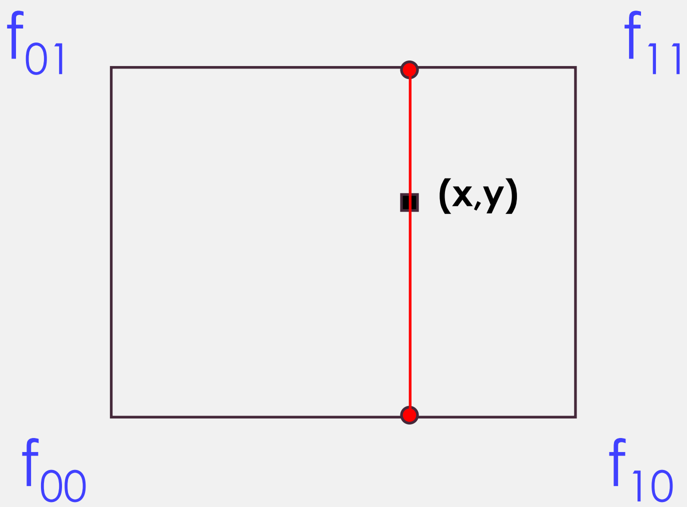
Bilinear Interpolation

- Consider one grid rectangle:
  - suppose corners are at (0,0), (1,0), (1,1), (0,1) ... ie a unit square
  - values at corners are f_{00}, f_{10}, f_{11}, f_{01}



How do we estimate value at a point (x,y) inside the square?

Bilinear Interpolation



We carry out three 1D interpolations:

(i) interpolate in x-direction between f_{00},f_{10}; and f_{01},f_{11}
(ii) interpolate in y-direction

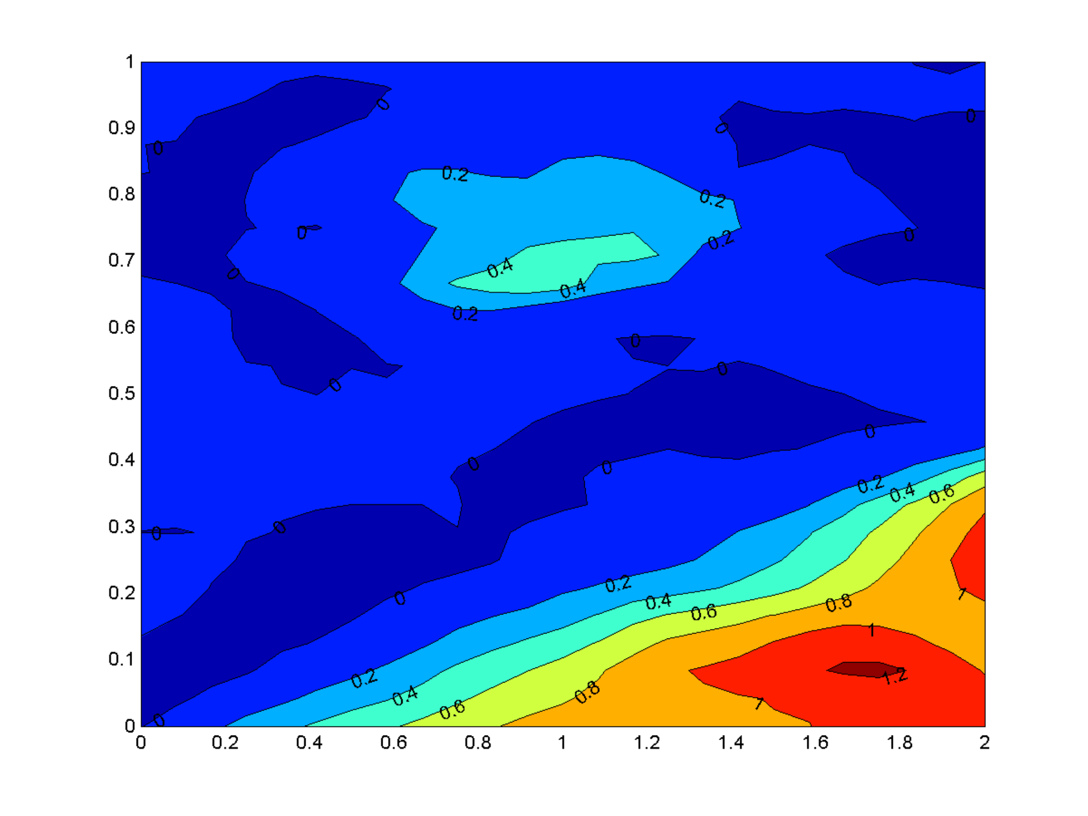
Exercise: Show this is equivalent to calculating -
$f(x,y) = (1-x)(1-y)f_{00}+x(1-y)f_{10}+(1-x)yf_{01}+ xyf_{11}$

Piecewise Bilinear Interpolation

- Apply within each grid rectangle
- Fast
- Continuity of value, not slope (C^0)
- Bounds fixed at data extremes

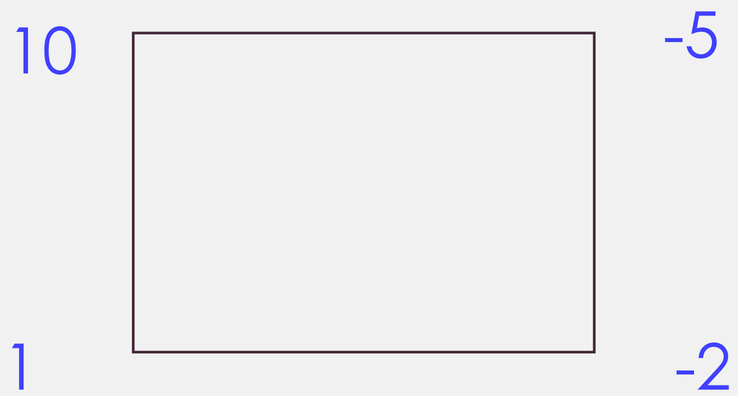
Contour Drawing

- Contouring is very common technique for 2D scalar data
- Isolines join points of equal value
  - sometimes with shading added
- How can we quickly and accurately draw these isolines?



An Example

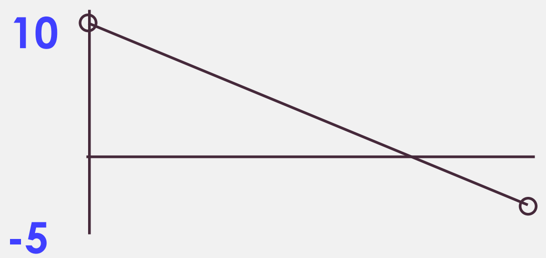
- **As an example, consider this data:**



Where does the zero level contour go?

Intersections with sides

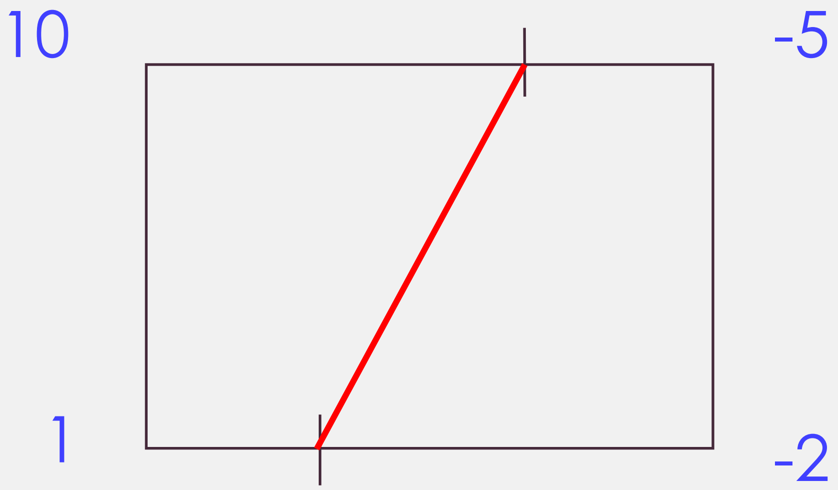
- The bilinear interpolant is linear along any edge - thus we can predict where the contour will cut the edges (just by simple proportions)



cross-section view along top edge

Simple Approach

- A simple approach to get the contour inside the grid rectangle is just to join up the intersection points



Question:
Does this always work?

Try an example where one pair of opposite corners are positive, other pair negative