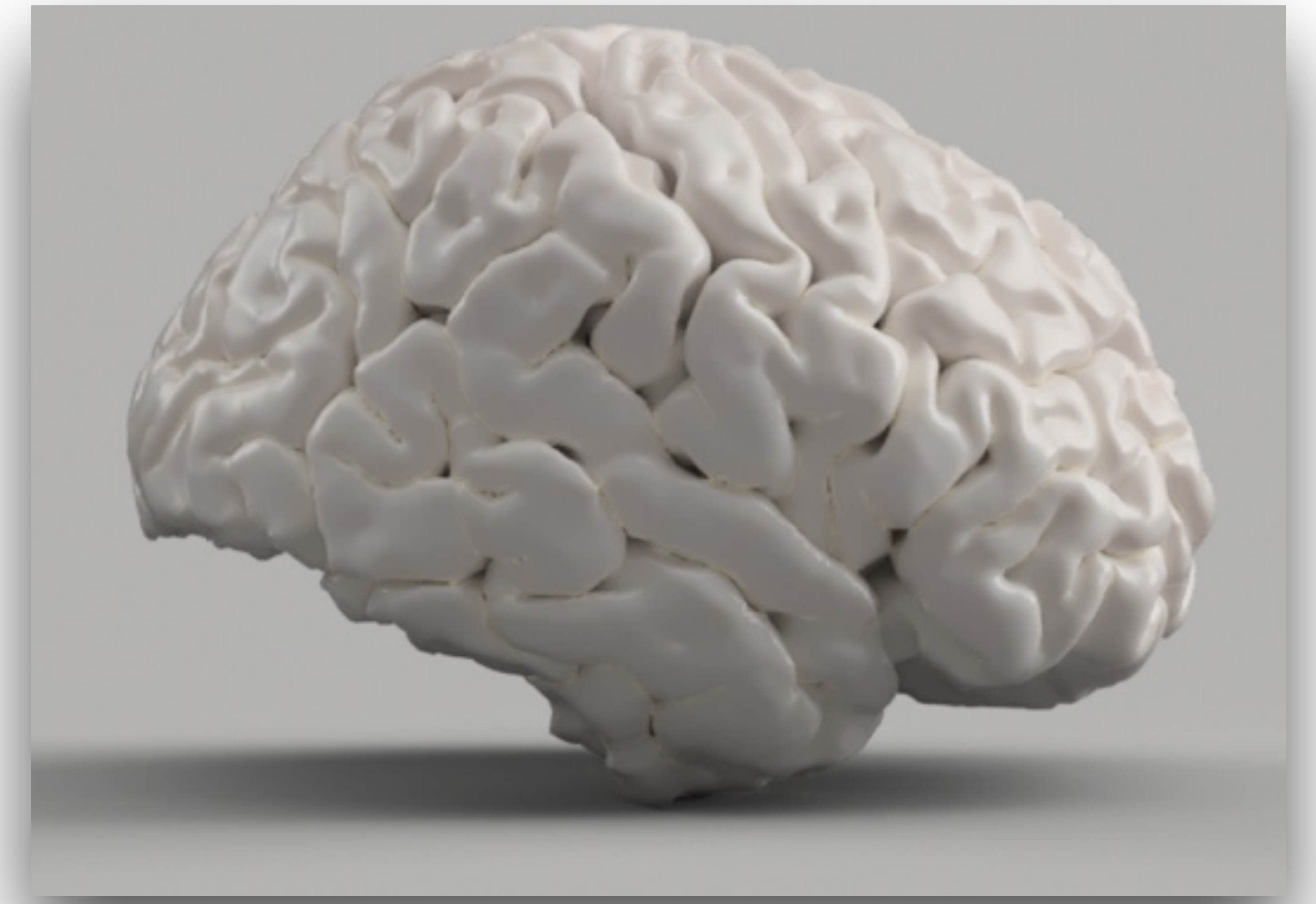
COGS109: Lecture 12



Regression II, Lagrange, Splines, error analysis

July 25, 2023

Modeling and Data Analysis

Summer Session 1, 2023

C. Alex Simpkins Jr., Ph.D.
RDPRobotics LLC | Dept. of CogSci, UCSD

Plan for the lecture

- Finish up discussion of least squares
    - Mention optimization and initial motivation
    - Error analysis
- Further curve fits (interpolation) - from linear to Lagrange to Splines

Upcoming deadlines

- http://casimpkinsjr.radiantdolphinpress.com/pages/cogs109_ss1_23/assignments.html
- Tonight Proposal/CP1
- Friday - A1, D4, Q3
- Sunday - CP2: EDA
- Next Tuesday A2, D5, D6

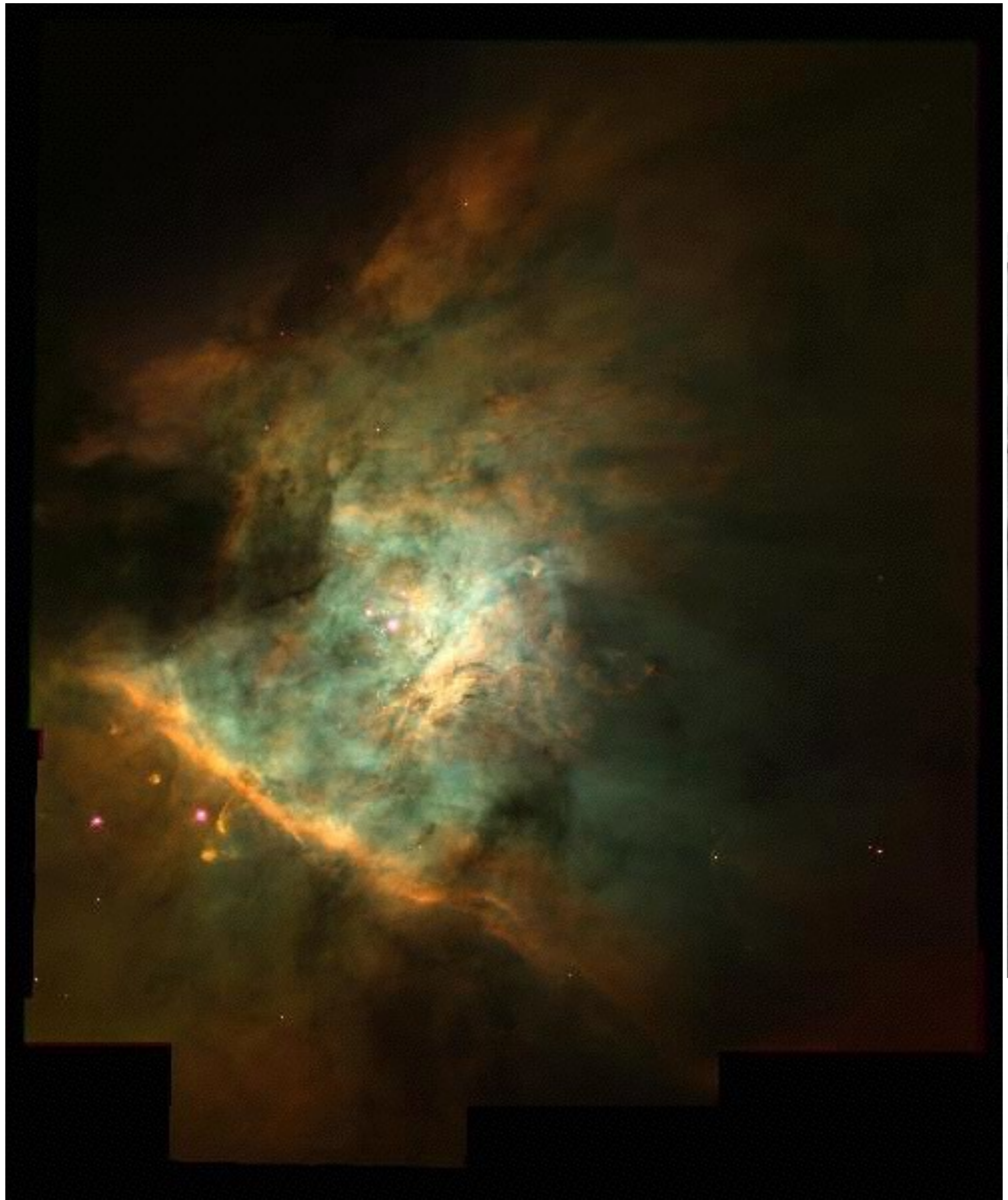
What if we want to fit something like this?

- This is not linear in the parameters, and it could be difficult to approach using linear least squares!

$$f(x) = x\left(ae^{bx} + sin(c\pi x)\right)$$

- Might want to ask - do we *reallllly* need all these terms like this?!?
- But don't worry, we have methods to approach this - ***Optimization***!
- Coming up in another lecture, just starting to motivate it

Another SDSC example: the orion nebula animation



Today we'll develop methods of nonlinear interpolation and extrapolation

- Lagrange
  - Useful for low number of data points
  - Unstable for high numbers of data points

- Splines
  - There are many kinds discussed in the reading, we'll just discuss one today
  - Good overall method
  - Works with many or few data points

Lagrange interpolation/extrapolation

- Fit a polynomial of degree that is the same as the number of points
  - If n points, degree of polynomial is n
- Makes a curve that exactly passes through all data points
- Use only for small number of data points

Lagrange derivation

- In nonlinear interpolation, we can fit an (n-1)st order curve exactly through n data points
- This is the lowest order curve, since an [n-1]st order polynomial will have exactly n parameters
- This technique is especially useful in cases with very few data points.
    - For large numbers of data points, above a few, it is more appropriate to use some form of cubic spline interpolation where a curve is fit through each pair of points.
    - Lagrange PDF

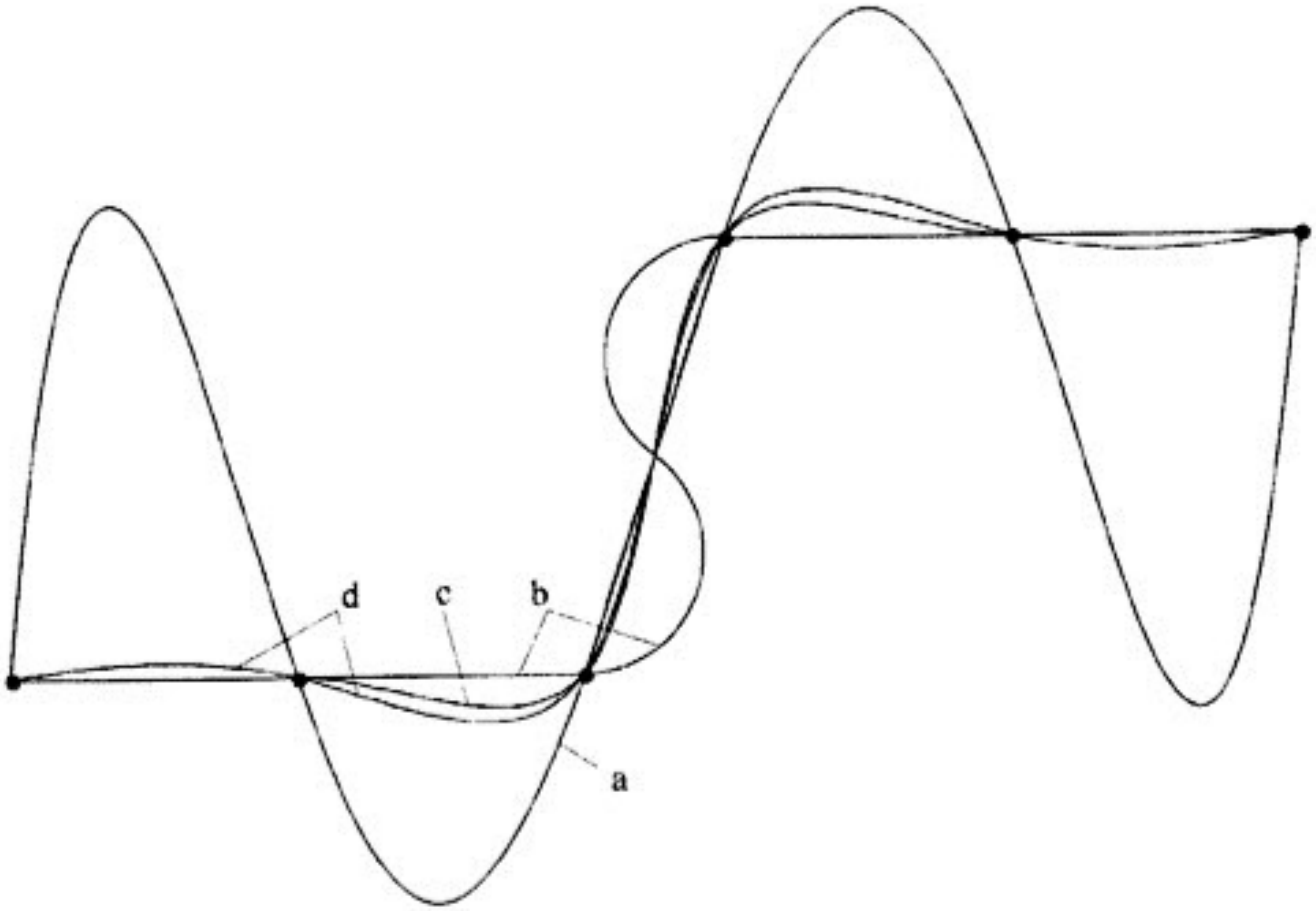
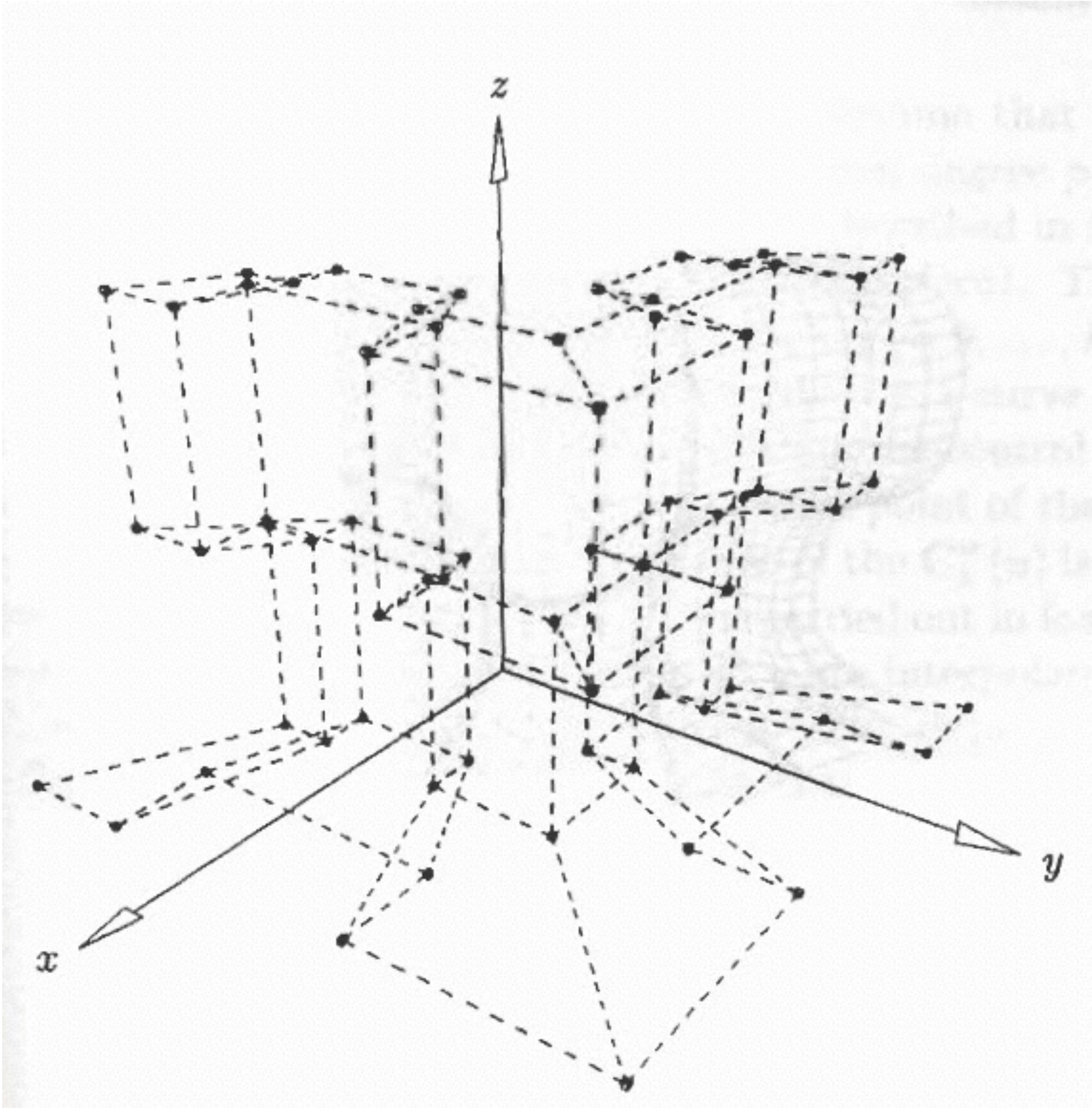


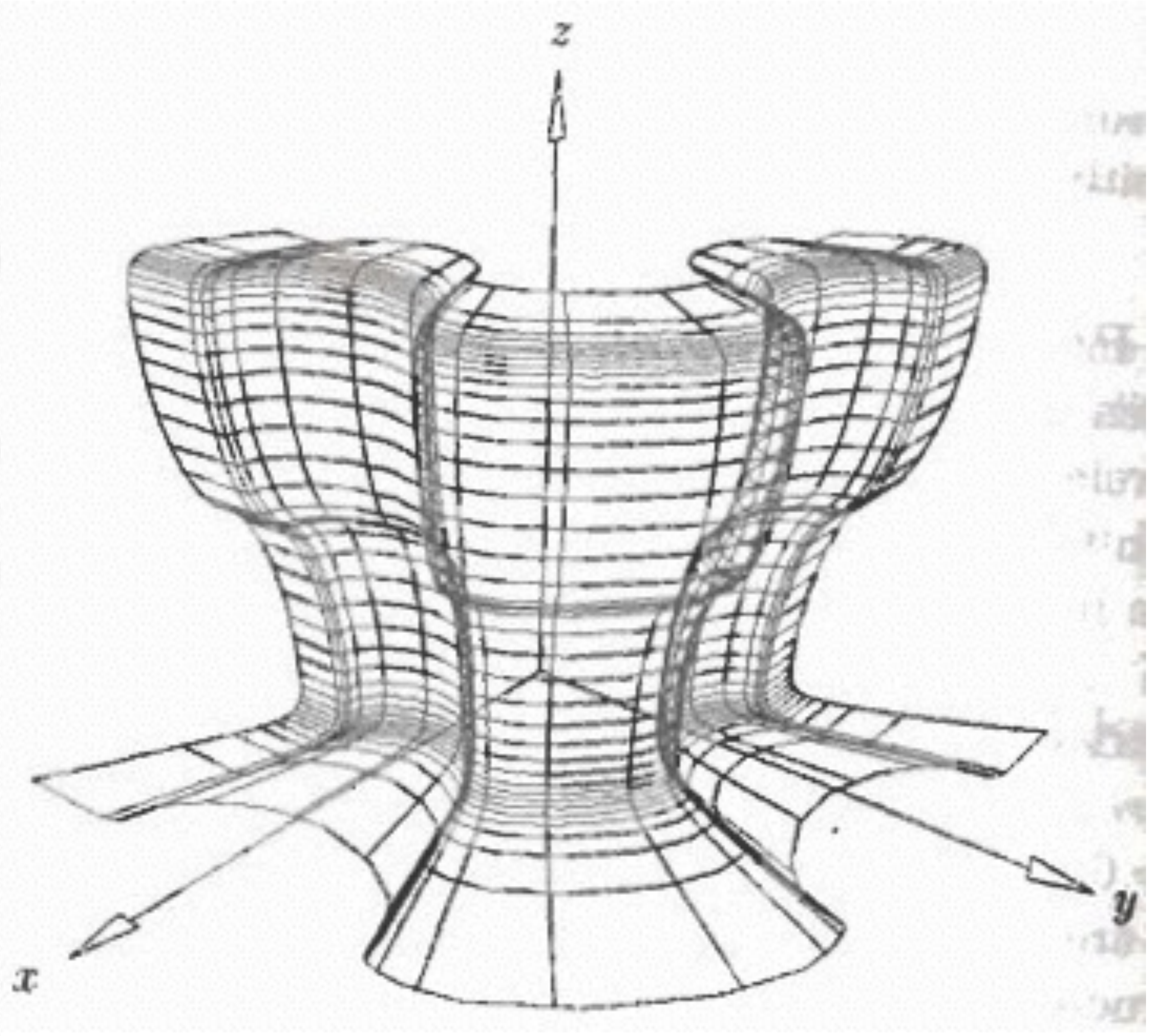
Figure 7.1 Interpolation curves drawn for six vertex points (dots), with y plotted in the vertical and x in the horizontal direction. Curves are shown for (a) a high-order polynomial fit, (b) a circular-arc fit, (c) a parabolic blend, and (d) a natural cubic spline.

Splines are useful in many places Lagrange fails

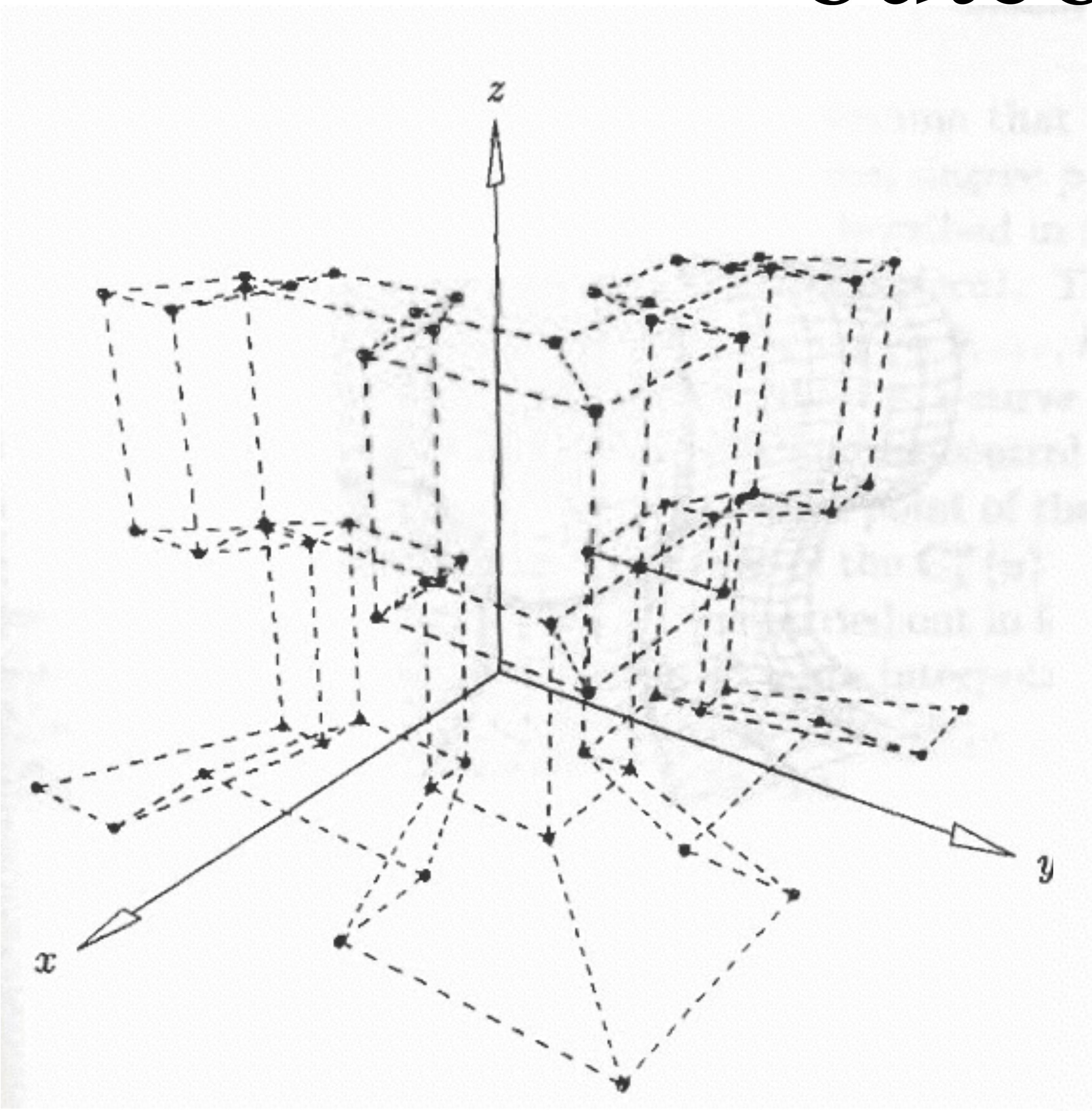
- Large number of data points
- Also can make a curve that passes through all data points
  - some types do not enforce this
- Drawn from drafting who drew from classical fine woodworking
  - Thin piece of wood stretched between pegs to create curves
  - Many types of splines dependent on end conditions
    - Pull tightly on the spline, curve gets sharper about the data points

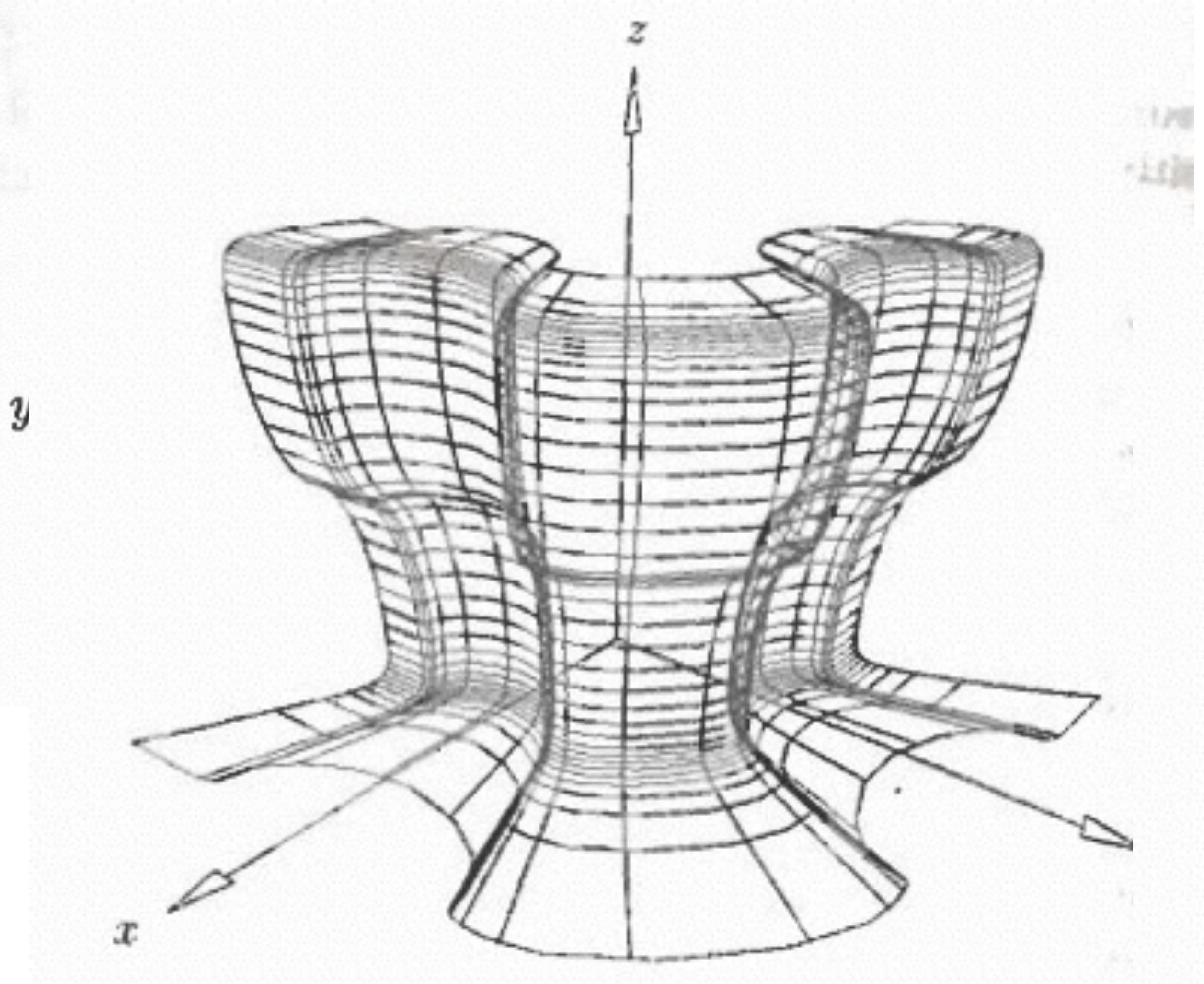
Splines are useful for N-Dimensions

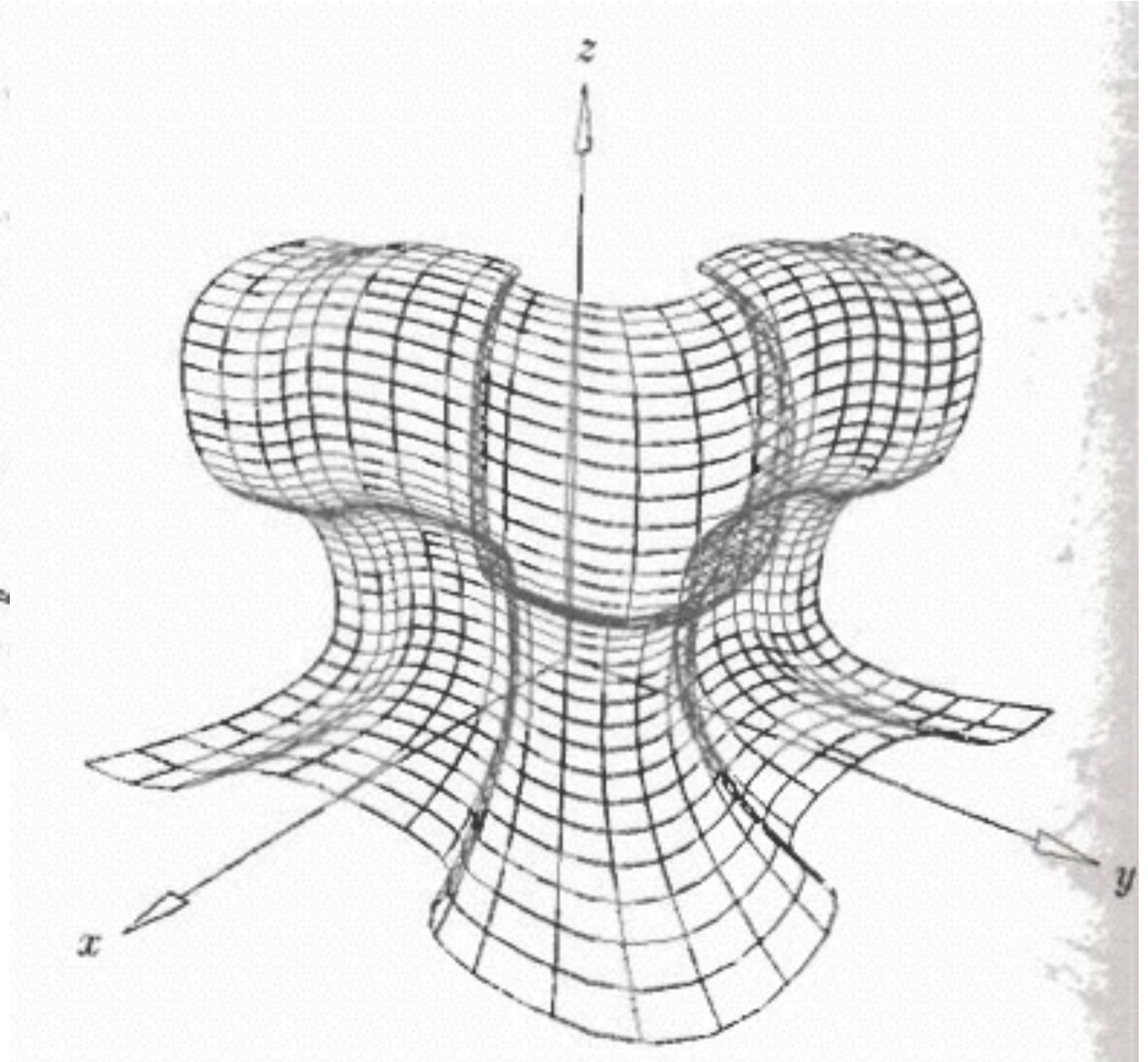




Splines also give you control over the final outcome of the curve







Some types of splines

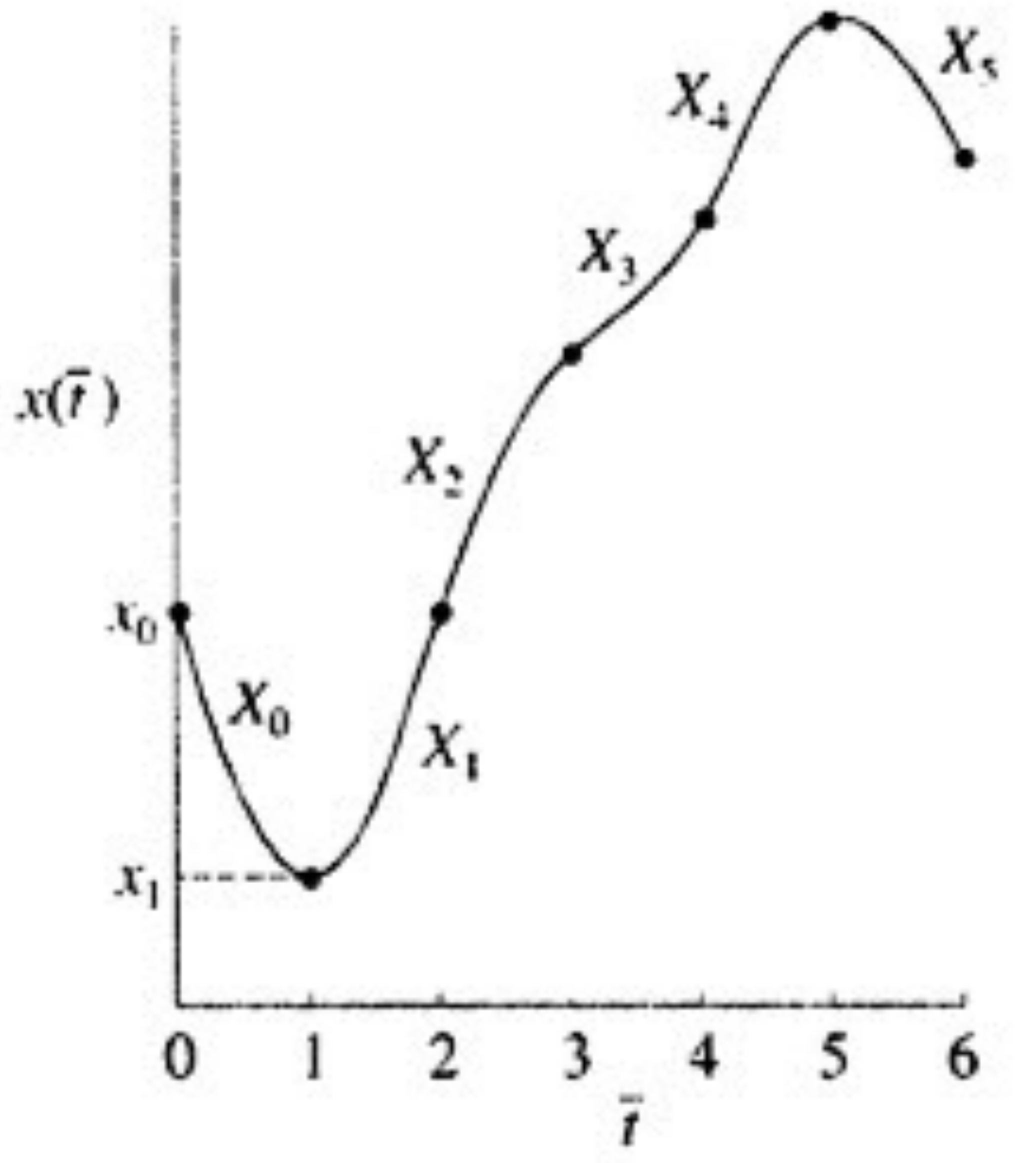
- Natural cubic spline
- Quadratic B-Splines
- Hermite Cubic Splines
- Coons Cubic Splines
- Rational B-Splines
- NURBS (Non-Uniform Rational B-Splines)

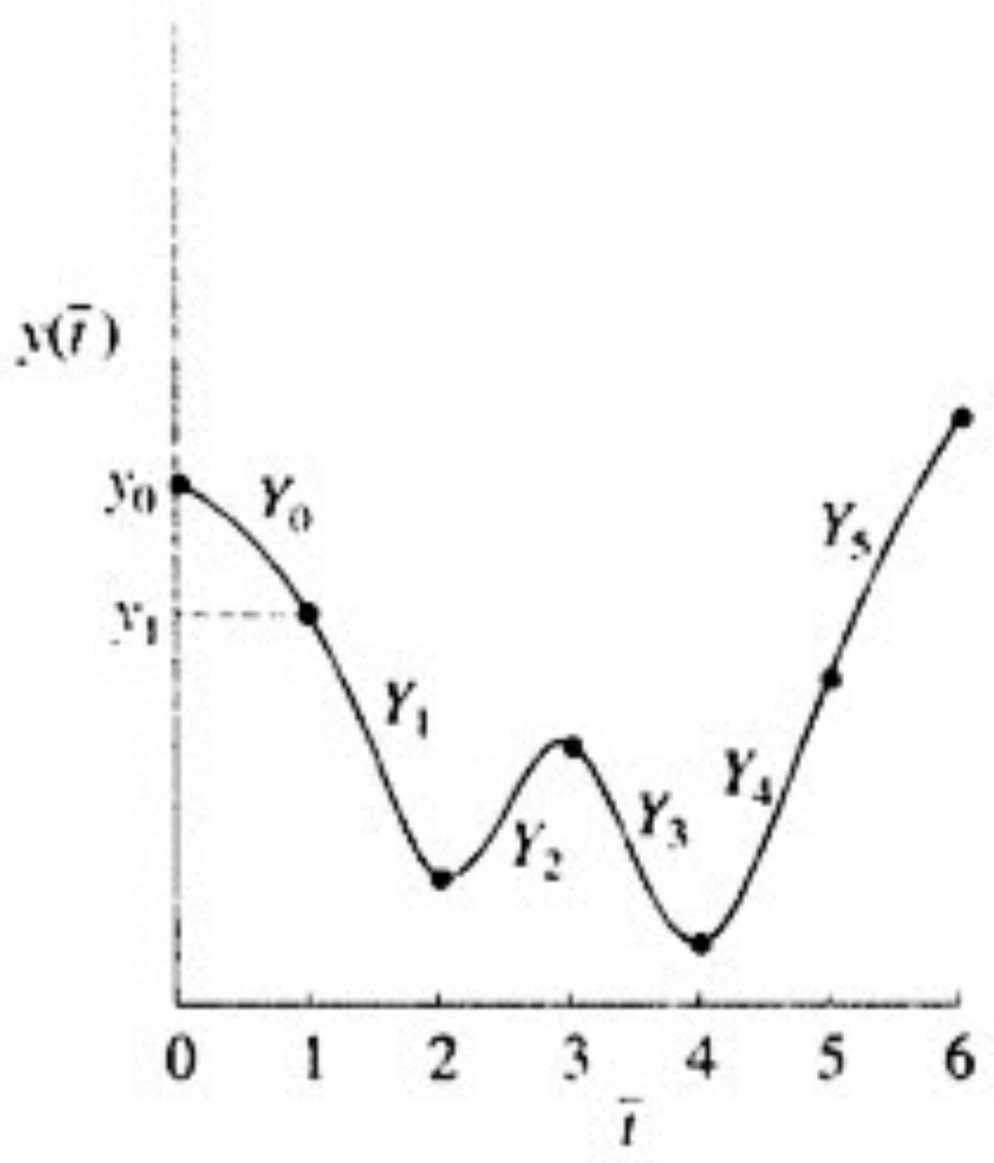
What we will discuss

- Natural cubic splines
  - **Why cubic?**
    - Because a curve is 'wiggly' and this is the lowest order polynomial that satisfies the conditions we're going to lay out
    - Higher order gets too oscillatory

Natural Cubic Spline - a conceptual introduction

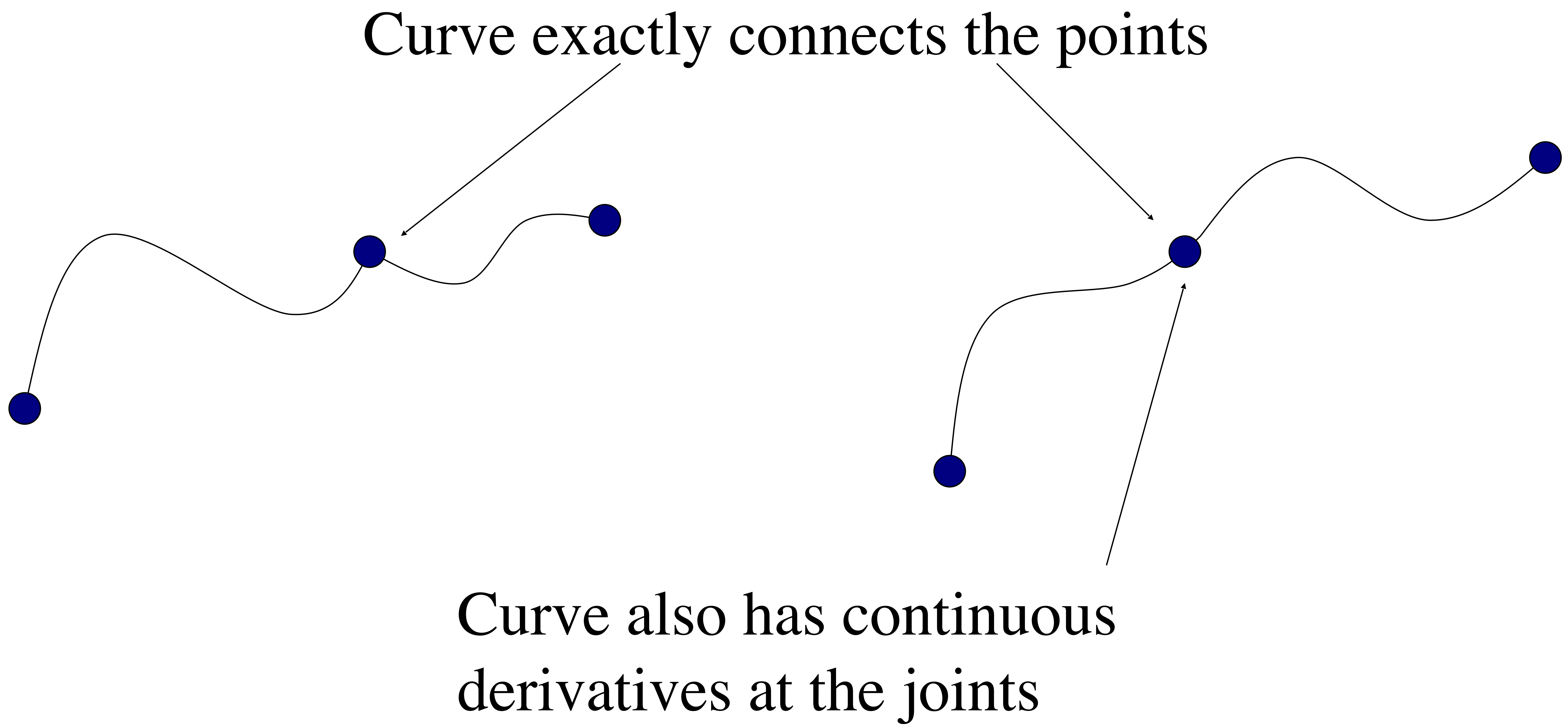
- We construct the following curves in sections





Adding constraints to solve for the unknowns

- Continuity at the joints:



Natural Cubic Splines

- We fit another parametric curve (similar to LERP), with a value of t from 0-1 again and make the ith segment according to

$$Y_i(t) = a_i + b_i t + c_i t^2 + d_i t^3$$

- And we solve for each set of these constants by requiring continuity at the end points (one section smoothly flows into the next, and the slope must match as well)

$$Y_i(0) = y_i = a_i$$

$$Y_i(1) = y_{i+1} = a_i + b_i + c_i + d_i$$

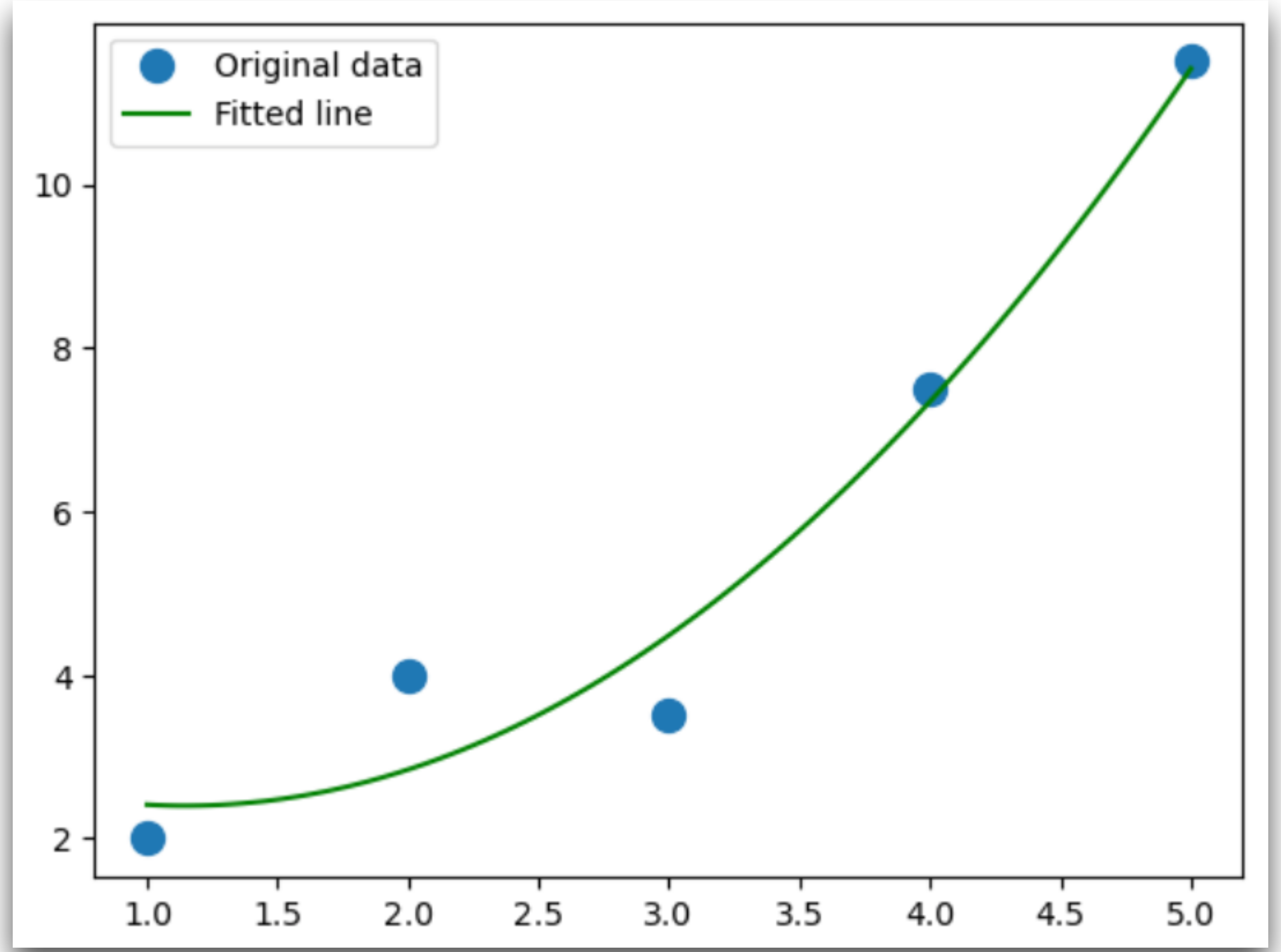
$$Y_i'(0) = D_i = b_i$$

$$Y_i'(1) = D_{i+1} = b_i + 2c_i + 3d_i$$

Back to error analysis

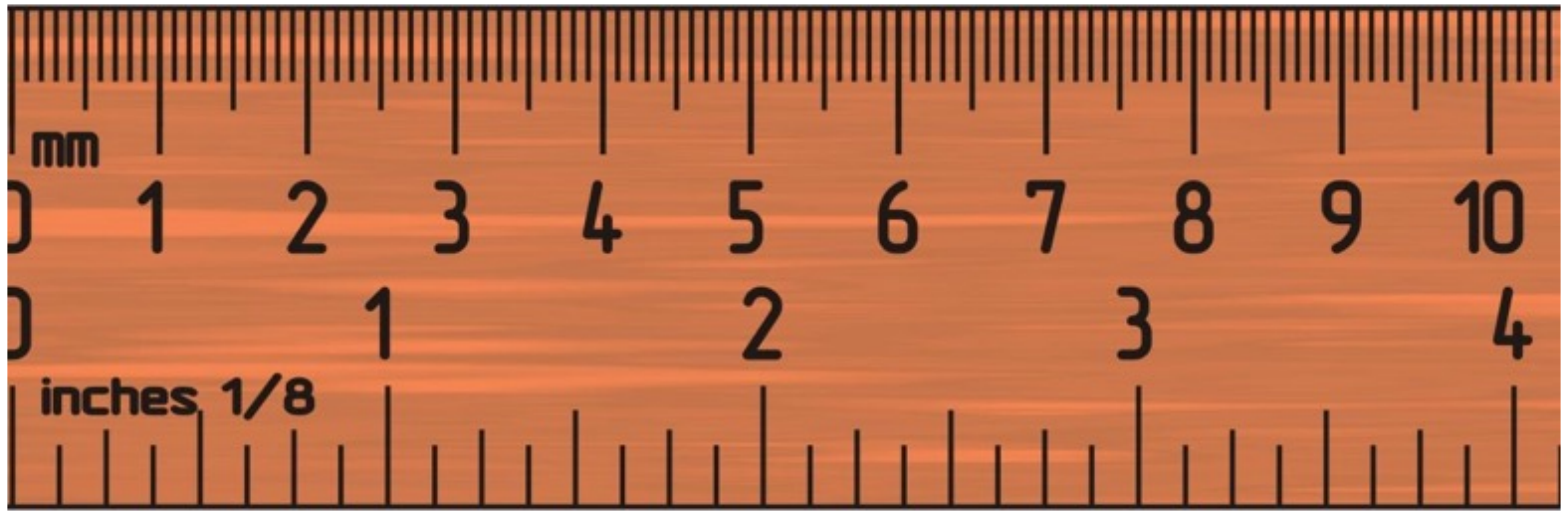
- We need to assess the quality of our fit

- Is this any ‘good?’



Uncertainty

- Error does not mean, in science, mistake
  - **It means the level of uncertainty in measurements and calculations**
  - **Can't eliminate by being careful, must instead minimize them**
- Basically want to have an estimate which is as reliable as possible
  - **'keep an eye on' your uncertainty**



Impossibility of certainty

- No physical quantity can be measured with absolute certainty
  - Wood door example
- Mathematical approximations to real systems are ALWAYS approximations, no matter how good
  - Any model you make is ONLY an approximation and should NEVER be confused with the real system
    - **Wrong**: "The Brain is computing the inverse of this matrix"
    - **Right**: "Our model approximates what the brain is doing by computing the inverse of this matrix"

The question…

- The question is not whether you are right or not
- The question is whether your approximation is good enough to be useful, dependent on what you consider to be ‘good enough’

Computing the estimated error

- One way to assess how good your model is consists of computing an estimated error
  - Typically you then decide whether your error is ‘within bounds’
    - (you create a boundary, such as the error in measuring/predicting position of a limb in space must be less than 10 inches)

- Uses one of many possible methods

Different error estimates

- There are many ways to estimate errors, here are a couple of common ones
  - To get a single # - can use various norms $\|e\|_2 = \sqrt{\sum_i (y_i - \hat{y}_i)^2}$
    - 2-norm
    - Mean-squared-error $MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$
  - Curve - simple error (for a time dependent signal y(t)) $e(t) = y(t) - \hat{y}(t)$
  - Curve - prediction error $e_p(t) = y(t) - \hat{y}(t \mid t-1)$

We've already developed several models and methods!

- Least squares is a very common method of fitting a model
  - **Works by minimizing an estimate of modeling error**
  - **Linear model, nonlinear model**
- Linear and nonlinear methods of interpolation are models of data approximation
  - **Computes curves which exactly pass through data points or use the data to control aspects of the curve**
  - **LERP, BERP, TERP, SLERP, Splines and lagrange**

Different ways of modeling based on data

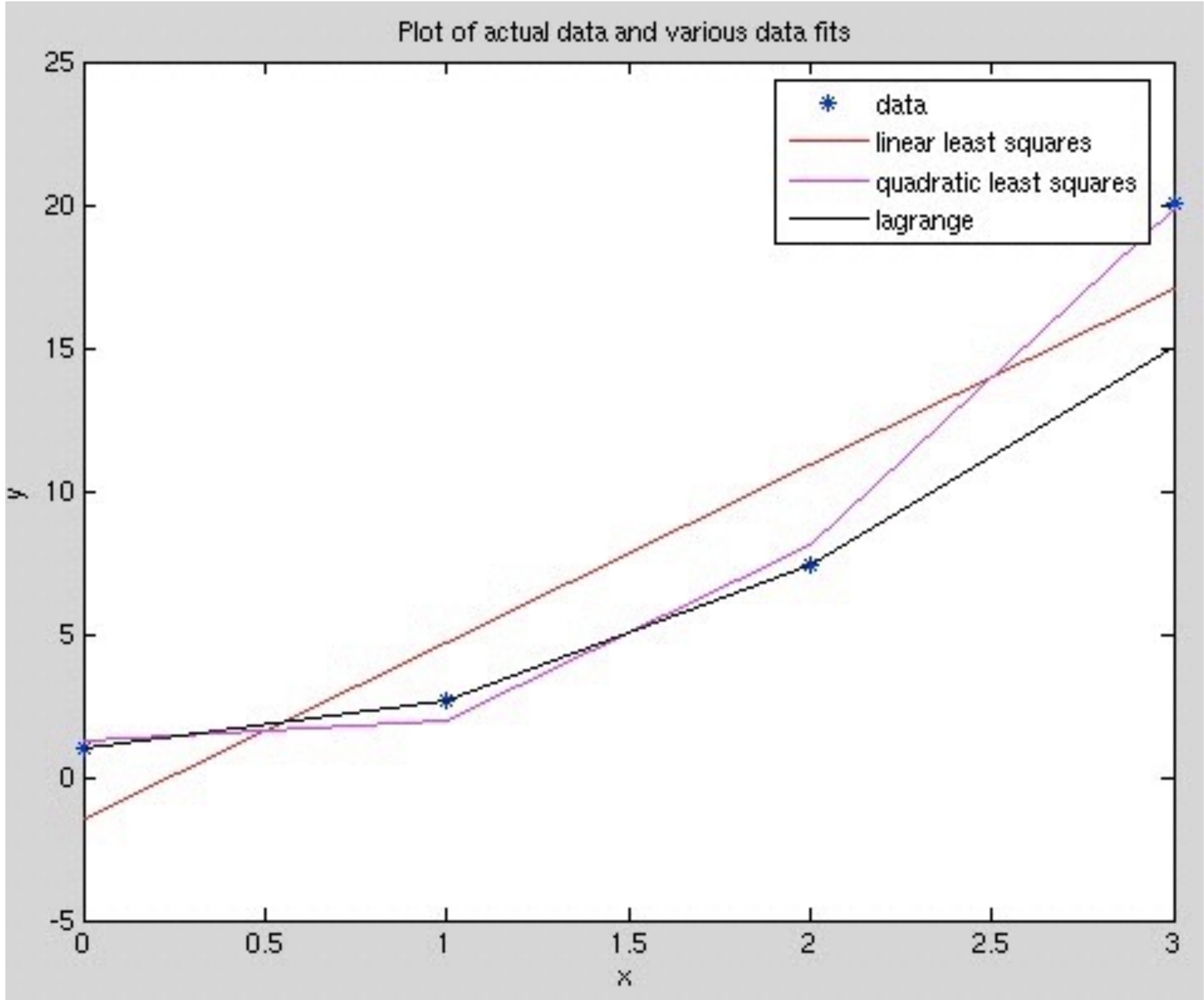
- Record all your data, then create a fit and study the resulting model
- Record all your data, split the recorded data into different groups
  - **use one group to fit a model**
  - **then the other to check and see how well does your model predict what the system does (this is called model validation - or ‘invalidation’)**

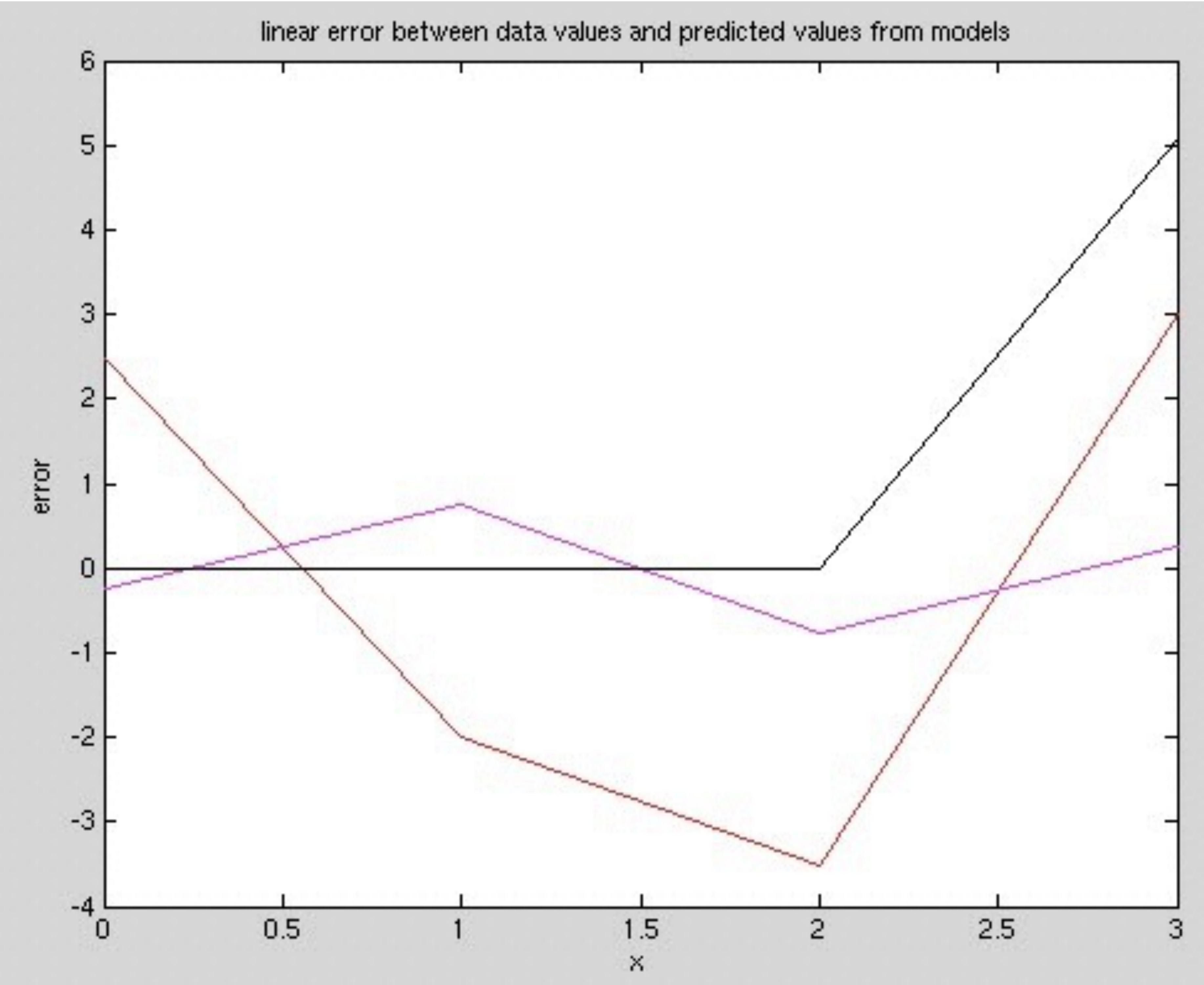
Example from the last few classes of computing error

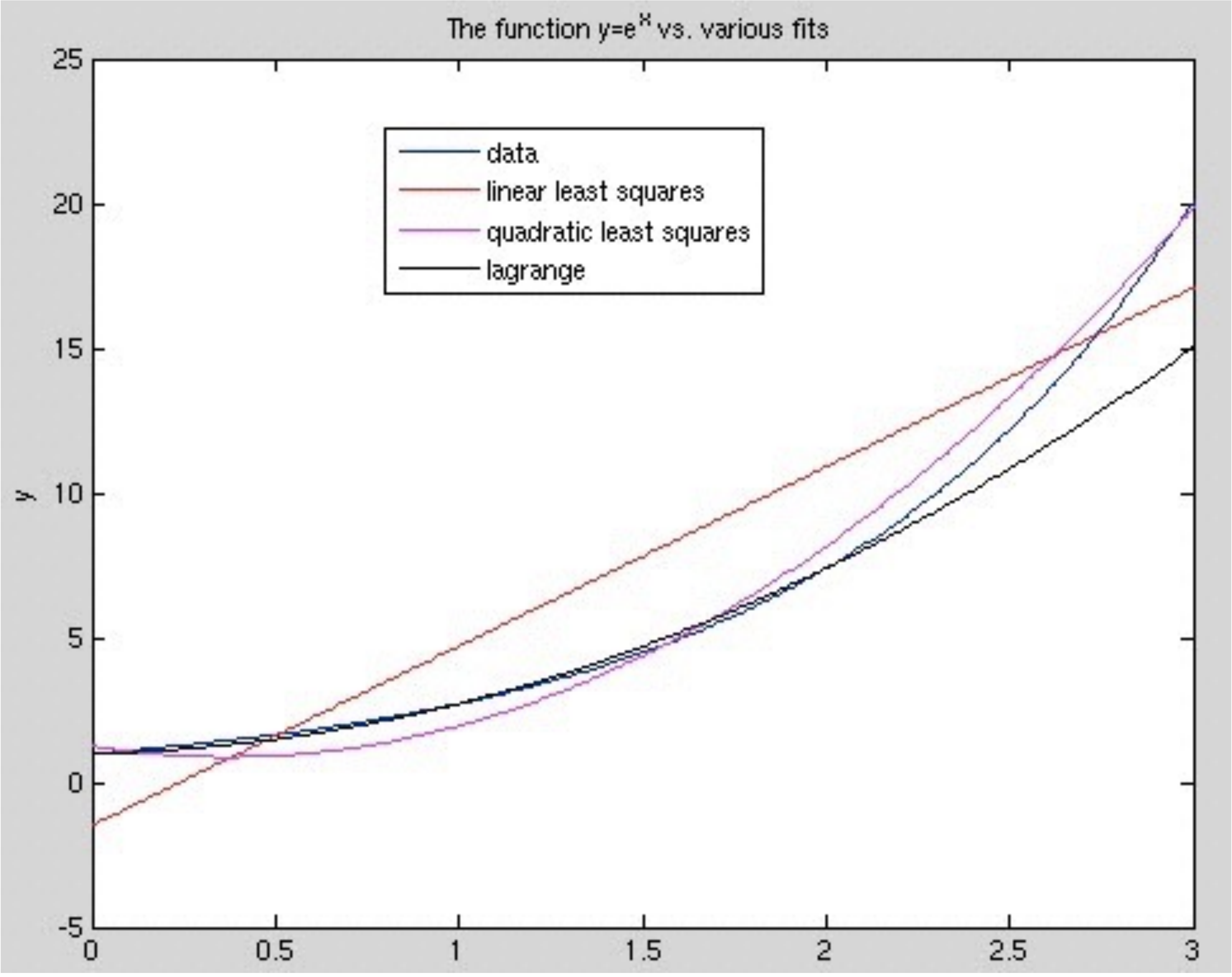
- Approximation of $y = e^x$ using the various methods we know already
- I generated simulated data by computing y=exp(x) for a domain of [0, 1]at three points (0.0, 0.5, 1.0)
  - **Then I created a linear least squares fit, a quadratic least squares fit, and Lagrange fits**

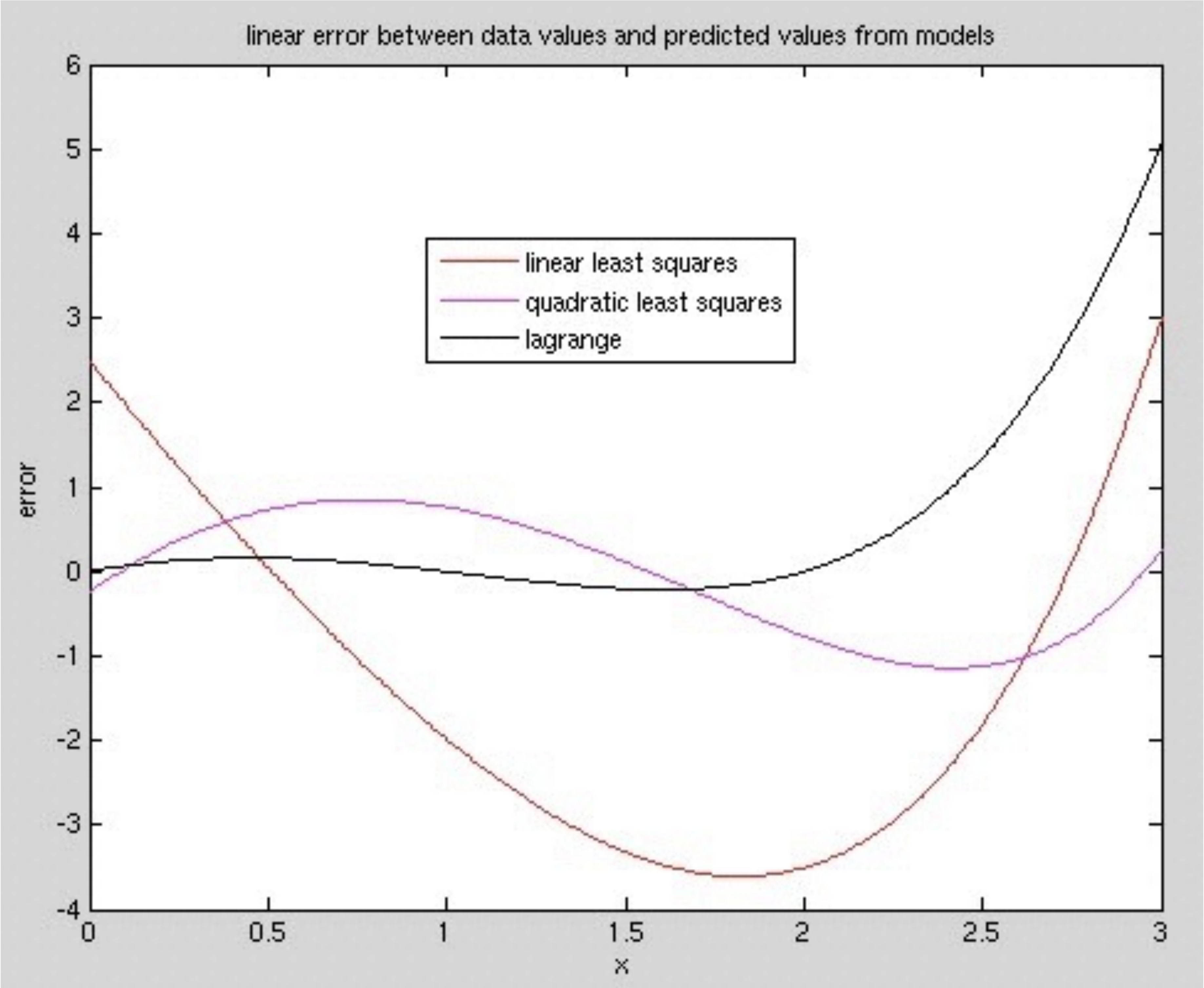
Assessing the models

- I assess how well each model fit does by first plotting the error between the data and the different methods
- Then I plot the real function (or data) vs. the different methods along a continuous curve


Plot of actual data and various data fits
data
linear least squares
quadratic least squares
lagrange
y
x


linear error between data values and predicted values from models
error
x


The function y=e^x vs. various fits
data
linear least squares
quadratic least squares
lagrange
y
25
20
15
10
5
0
-5
0
0.5
1
1.5
2
2.5
3


linear error between data values and predicted values from models
linear least squares
quadratic least squares
lagrange
error
x
6
5
4
3
2
1
0
-1
-2
-3
-4
0
0.5
1
1.5
2
2.5
3

We can also compute the error as a single quantity

- e2lls = 125.7192
- e2nlls = 10.9367
- e2lag = 86.3331
- From this we see that over this interval, the nonlinear least squares polynomial fits the data the best if we're trying to minimize this error as a criterion for goodness of fit
- Again it depends on our criterion, as the lagrange has the lowest error over the domain of data used for computing the fit
  - **It doesn't extrapolate the future points as well in this case**