Introduction to Interpolation

CS370 – May 9, 2016

Round v.s. Truncation - Clarification

Floating point systems offer different *rounding modes*.

We considered:

1) **Round-to-nearest** – $fl(x)$ rounds to closest available number in F.
   - Usually the default.
   - We'll break ties by simply rounding ½ *up*. (Other options exist.)

2) **Truncation/Chopping** – $fl(x)$ rounds to next number in F *towards zero.*
   - i.e. discard any digits after the t^{th}.

Assignment #1 Posted

- See Piazza site, Resources page.
- Due May 26, 2016 @ 4pm. Submit to A1 Dropbox on Learn.

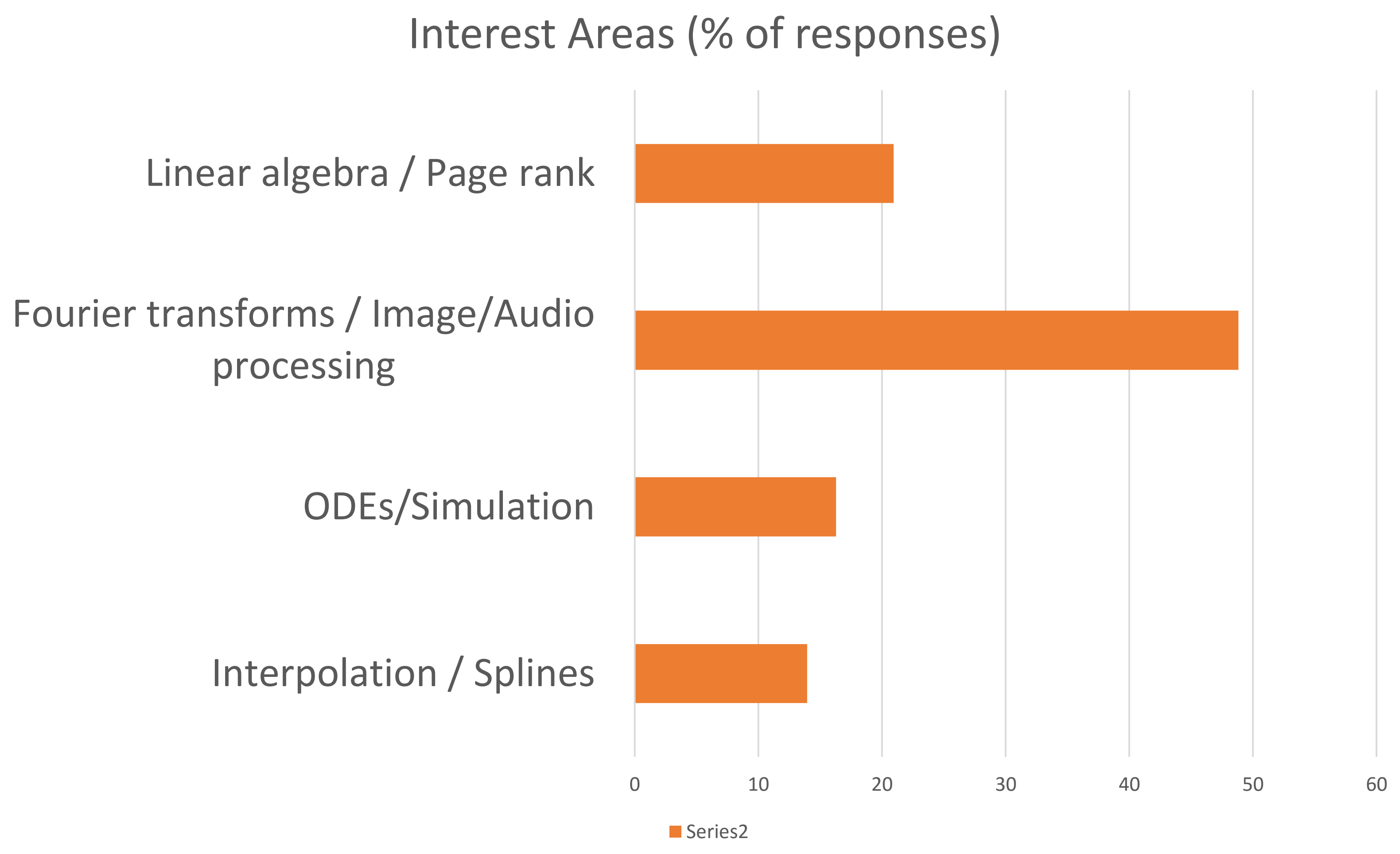
Reminder:

- Submit written work and all results/output as a single PDF.
- Submit Matlab code as a single ZIP file.

Matlab Tutorial Reminder

Tuesday, May 10 (tomorrow!) at 6:00pm, in MC1056, lead by TA Ke Nian.

Rough breakdown of interest per topics



The Basic Problem of *Interpolation*

Given a set of data points from an (unknown) function $y = p(x)$, can we approximate p's value at other points?

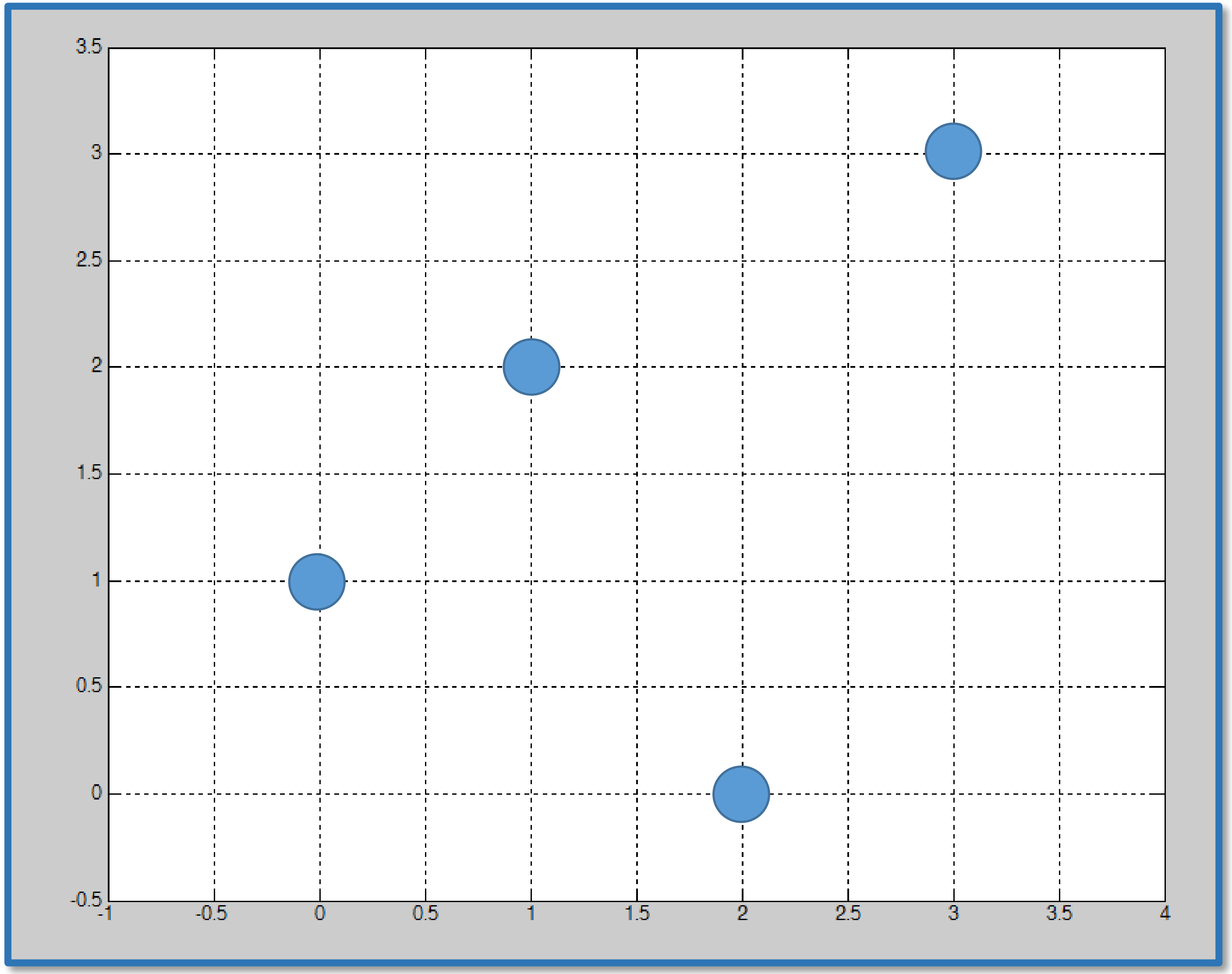
e.g., Given $p(x_1) = y_1, p(x_2) = y_2, \ldots, p(x_n) = y_n$.

Estimate $y = p(x)$ for any point x such that $x_1 \leq x \leq x_n$.

Visualized:

E.g. given points (x_i, y_i):
$(0,1), (1,2), (2,0), (3,3)$.

Find a function $p(x)$ that goes *exactly through* or ***interpolates*** all the points.

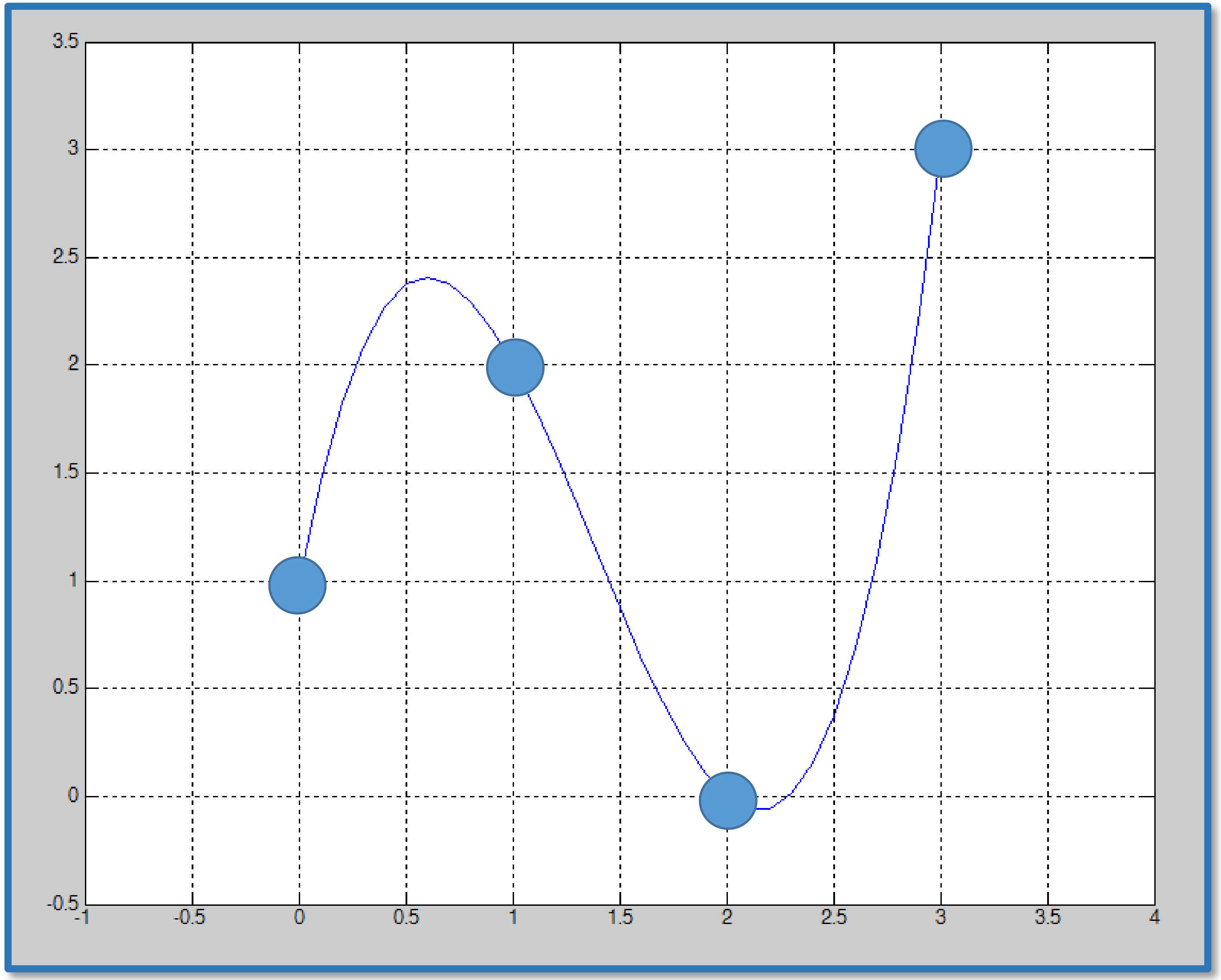


One Solution

$$y = \frac{4}{3}x^3 - \frac{11}{2}x^2 + \frac{31}{6}x + 1$$

Can now approximate $y = p(x)$ for other x. E.g., $p\left(\frac{1}{2}\right) = 2.375$.

This interpolating function ("interpolant") is not *necessarily* unique! Many other functions also hit these points.

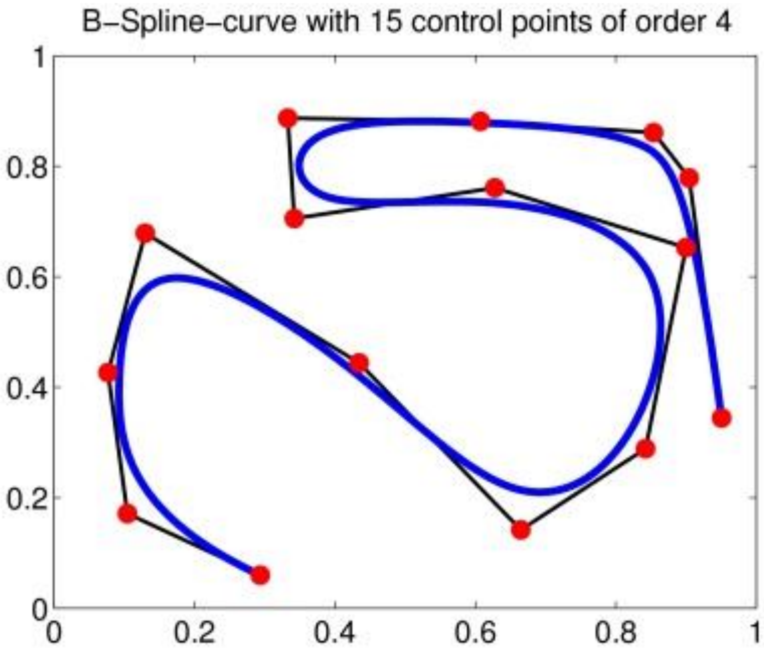


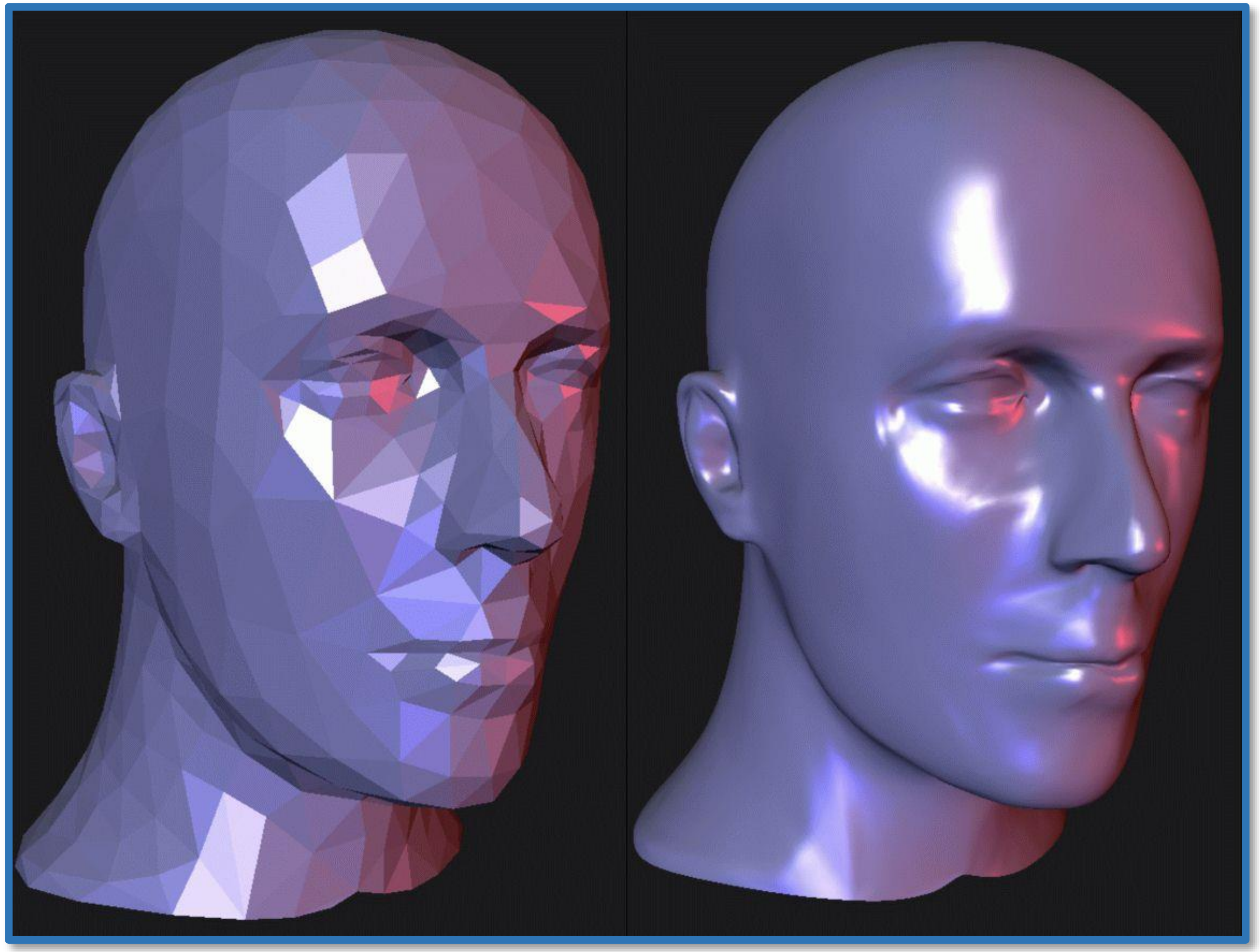
Interpolation – Uses

- Fitting curves to data. (Related to regression in statistics/ML.)
- Estimating an unknown function's properties: values, derivatives, etc.
- Plays a role in numerical methods for:
  - differentiation
  - integration
  - differential equations
  - optimization
  - lots more

Practical Examples – 2D/3D Design

By editing a smaller number of ***control points***, a smoother surface that *interpolates* those points can be constructed.



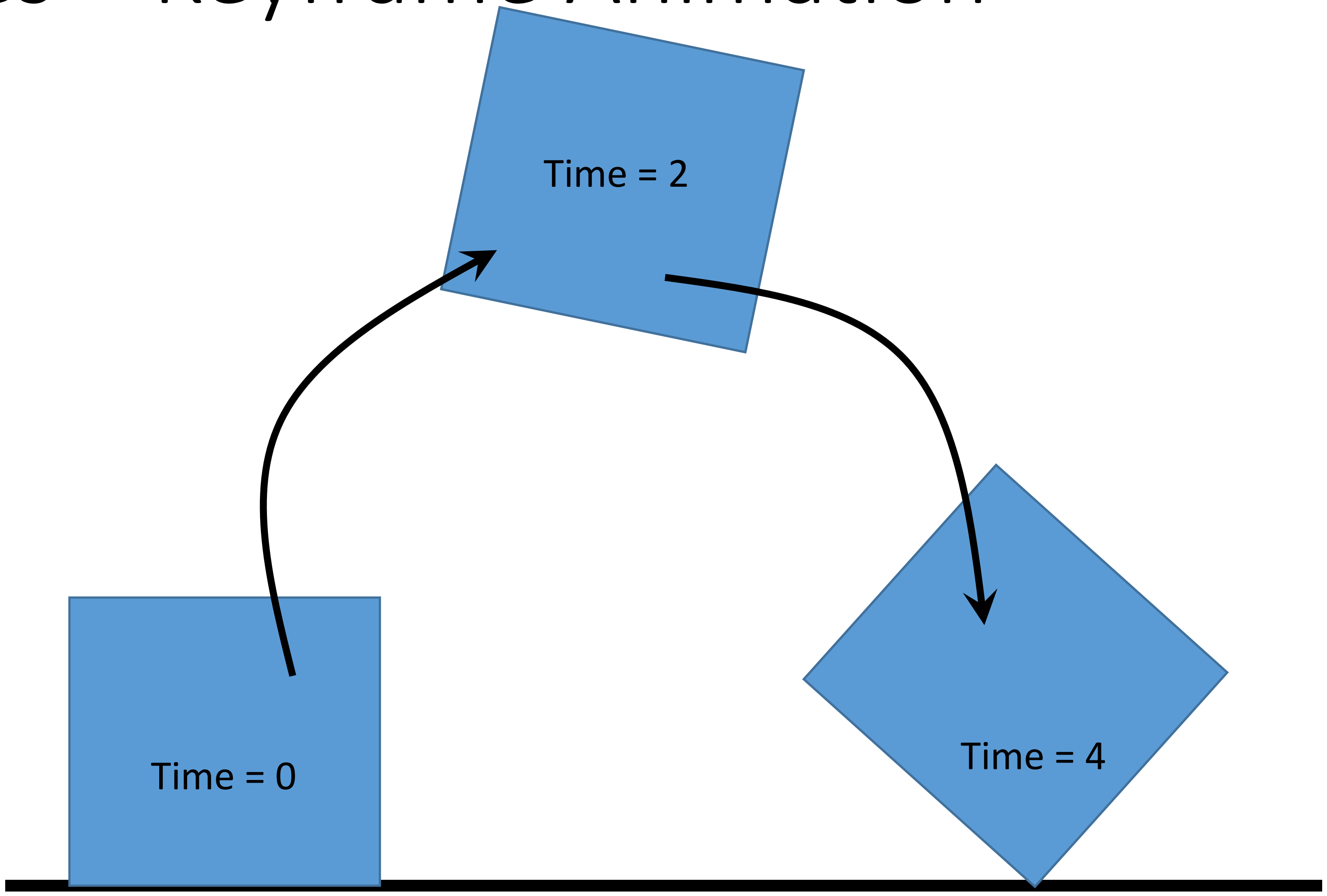


Low Detail Input Mesh w/ Sharp Corners

Smooth Interpolated Surface

Practical Examples – Keyframe Animation

By placing only a few ***keyframes***, a smooth motion path for an object (or character) is constructed that *interpolates* those points.



Keyframing Demo (Blender)

- Blender: an open-source 3D modeling/animation/rendering package.
- Let’s see keyframing in action, and consider how interpolation comes into play.

Interpolation Overview

We'll begin with methods for interpolating few points (often ≤ 6).

- Polynomial Interpolation
  - Vandermonde matrices
  - Lagrange form

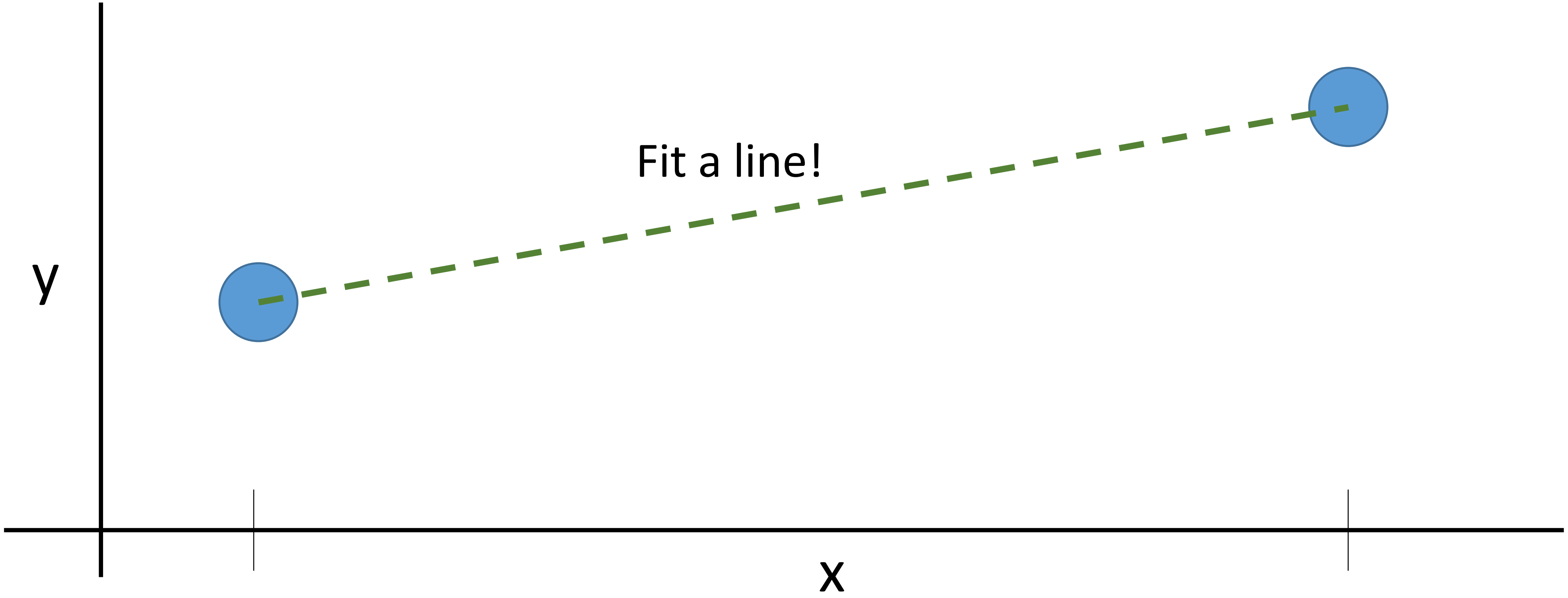
Mostly today

Later, interpolation for many points.

- Piecewise interpolants:
  - Piecewise linear
  - Cubic splines
  - B-splines, Bezier curves (time permitting)

Simplest Problem – Linear Interpolation

Given just two points, how might we approximate points that lie in between?



Fitting a line to two points

Write down the line equation $y = ax + b$ for unknowns, a and b.

For 2 points we have exactly 2 equations & 2 unknowns.

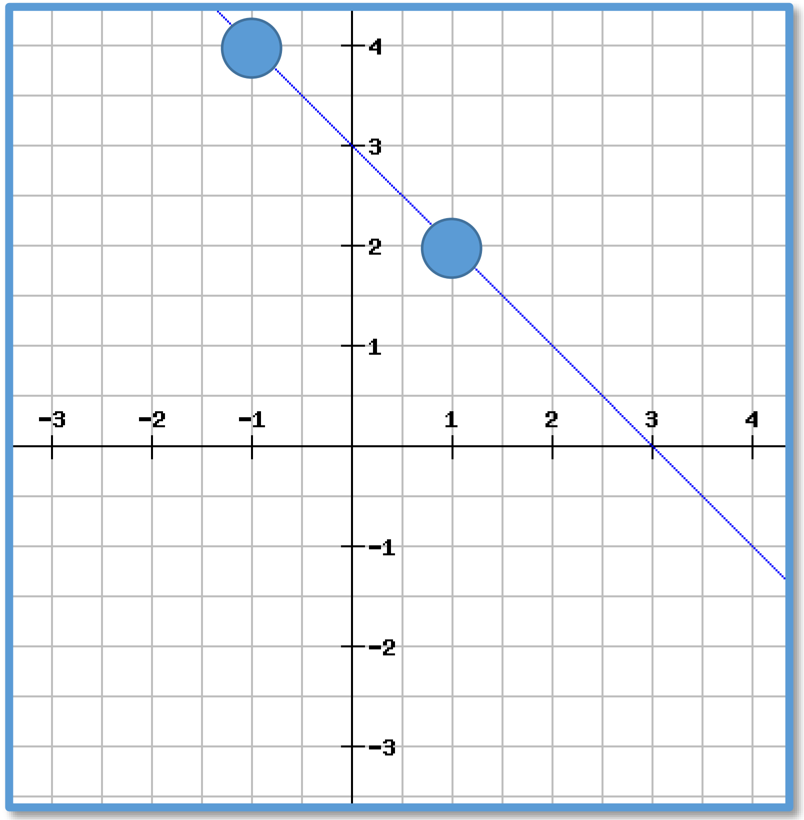
Example: Given $(x_1, y_1) = (1,2), (x_2, y_2) = (-1,4)$, find a and b for the line passing through the points.

Try to work it out!

Line-fitting solution

Input points were:

$$(x_1, y_1) = (1,2)$$
$$(x_2, y_2) = (-1,4)$$



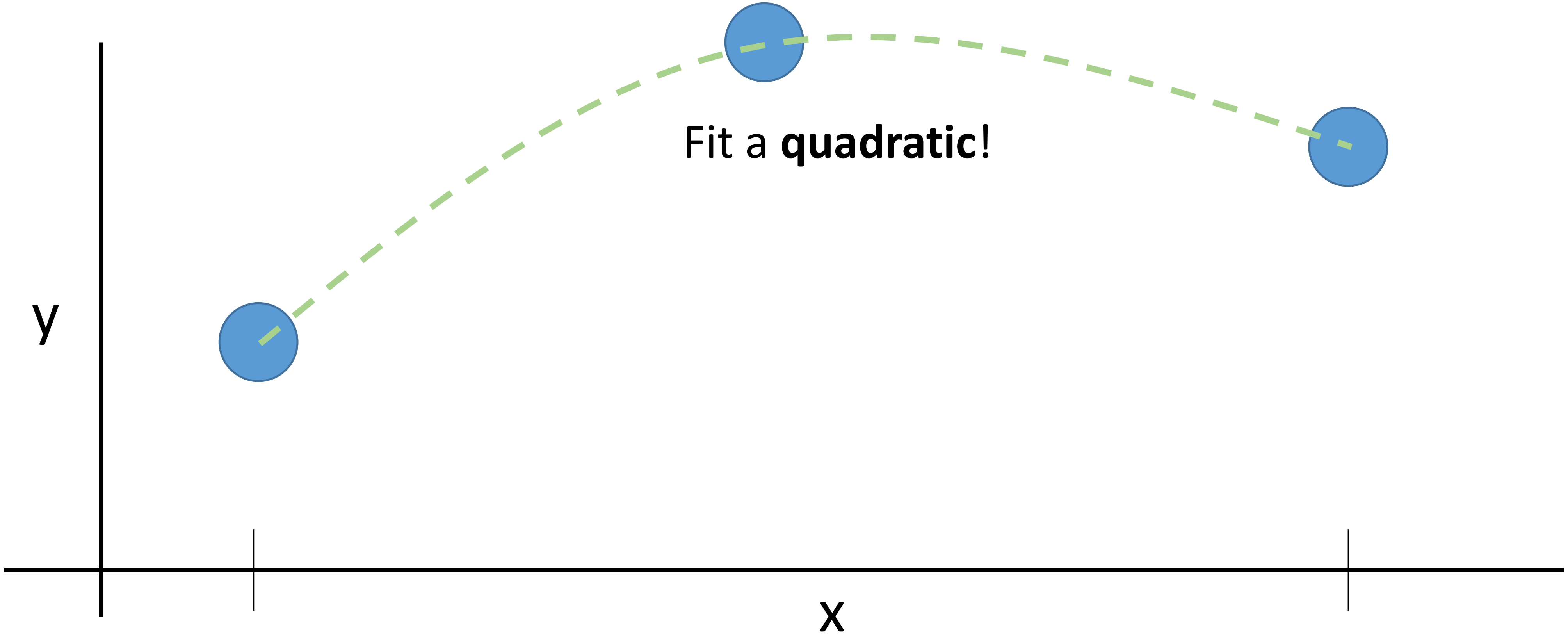
We found the line

$$y = -x + 3.$$

It does indeed *interpolate* the points.

Adding A Third Point

Given just three points, how might we approximate points that lie in between?



Fitting a quadratic to 3 points

For 3 data points, we find the 3 unknown coefficients, a, b, c, for a quadratic…

$$y = ax^2 + bx + c.$$

Each data point gives 1 *linear* equation, so we get a 3x3 system:

$$\begin{aligned} ax_1^2 + bx_1 + c &= y_1 \\ ax^2 + bx_2 + c &= y_2 \\ ax_3^2 + bx_3 + c &= y_3 \end{aligned}$$

or

$$\begin{bmatrix} x_1^2 & x_1 & 1 \\ x_2^2 & x_2 & 1 \\ x_3^2 & x_3 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

Polynomial Interpolation

Can we generalize to arbitrarily many points?

$$p(x) = c_1 + c_2 x + c_3 x^2 + c_4 x^3 + \cdots + c_n x^{n-1}$$

Yes we can!

Unisolvence Theorem:

Given n data pairs $(x_i, y_i), i = 1, \dots, n$ with *distinct* x_i, there is a unique polynomial $p(x)$ of degree $\leq n - 1$ that interpolates the data.

When is the degree less than $n - 1$?

Polynomial Interpolation

For n points, must find all the coefficients c_i of the polynomial

$$p(x) = c_1 + c_2 x + c_3 x^2 + c_4 x^3 + \cdots + c_n x^{n-1}.$$

As before, each (x_i, y_i) point gives one *linear* equation

$$y_i = c_1 + c_2 x_i + c_3 x_i^2 + c_4 x_i^3 + \cdots + c_n x_i^{n-1}.$$

Then solve the $n \times n$ linear system.

Example: Fitting General Polynomials

Very first example had 4 (x_i, y_i) pairs: $(0,1), (1,2), (2,0), (3,3)$.

What is the linear system needed to recover the coefficients of the *cubic* polynomial?

$$y_i = c_1 + c_2 x_i + c_3 x_i^2 + c_4 x_i^3$$

Vandermonde matrices

In general, we get a linear system:

$$\begin{bmatrix} 1 & x_1 & x_1^2 & \dots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \dots & x_2^{n-1} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & x_n & x_n^2 & \dots & x_n^{n-1} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ \dots \\ c_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y\_n \end{bmatrix}$$

Or…

$$V\vec{c} = \vec{y}$$

V is called a *Vandermonde matrix*.

Polynomial interpolation reduces to solving systems of equations.

Theoretical Implication

Properties of Vandermonde matrices, V, can be used to prove the unisolvence theorem.

If there is always a unique solution, then there is also a unique corresponding polynomial. So need to show that V is non-singular, i.e., $\det V \neq 0$.

One can use the fact that $\det V = \prod_{i<j}\left(x_i - x_j\right)$ (which can be proved by induction).

The *monomial* basis

The familiar form $p(x) = c_1 + c_2 x + c_3 x^2 + c_4 x^3 + \cdots + c_n x^{n-1}$ is called the monomial form, and can also be written

$$p(x) = \sum_{i=1}^{n} c_i x^{i-1} \, .$$

The sequence $1, x, x^2, x^3$ ... is called the *monomial basis*.

Monomial form is a sum of coefficients c_i times these *basis functions*.

The *Lagrange* basis

A different basis for interpolating polynomials.

We will define the *Lagrange basis functions*, $L_k(x)$, to construct a polynomial as

$$p(x) = y_1 L_1(x) + y_2 L_2(x) + \cdots + y_n L_n(x) = \sum_{k=1}^{n} y_k L_k(x)\,.$$

where y_i are the coefficients (*and* also our data values,$y_i = p(x_i)$).

Lagrange basis functions and their properties

Notice: No x_k entry!

Given n data points (x_i, y_i), we define

$$L_k(x) = \frac{(x - x_1)(\ldots)(x - x_{k-1})(x - x_{k+1})(\ldots)(x - x_n)}{(x_k - x_1)(\ldots)(x_k - x_{k-1})(x_k - x_{k+1})(\ldots)(x_k - x_n)}$$

What is $L_i(x_j)$ for $i = j$? 1

Numerator & denominator are identical.

What is $L_i(x_j)$ for $i \neq j$? 0

One of the entries of the numerator is 0; denominator remains non-zero.

Lagrange polynomials interpolate the data

Therefore the $L_k(x)$ satisfy

$$L_i(x_j) = \begin{cases} 1, i = j \\ 0, i \neq j \end{cases}$$

What does this win us for a polynomial of the form

$$p(x) = y_1 L_1(x) + y_2 L_2(x) + \cdots + y_n L_n(x)?$$

Ensures $p(x)$ **must** interpolate each x_i, since

$$p(x_i) = y_1 L_1(x_i) + y_2 L_2(x_i) + \cdots + y_i L_i(x_i) + \cdots + y_n L_n(x_i)$$

$$p(x_i) = y_1 \cdot 0 + y_2 \cdot 0 + \cdots + y_i \cdot 1 + \cdots + y_n \cdot 0$$

i.e., $p(x_i) = y_i$ *by construction*.

Lagrange polynomials: Line example

$$p(x) = \sum_{k=1}^{n} y_k L_k(x)\,, \text{where } L_k(x) = \frac{(x - x_1)(\dots)(x - x_{k-1})(x - x_{k+1})(\dots)(x - x_n)}{(x_k - x_1)(\dots)(x_k - x_{k-1})(x_k - x_{k+1})(\dots)(x_k - x_n)}$$

Consider the two points we fit a line to earlier:

$$(x_1, y_1) = (1,2), (x_2, y_2) = (-1,4)$$

What are the corresponding L_k, and polynomial $p(x)$?

$$L_1(x) = \frac{x--1}{1--1} = \frac{x+1}{2}, \qquad L_2(x) = \frac{x-1}{-1-1} = \frac{x-1}{-2}$$

$$p(x) = 2L_1(x) + 4L_2(x) = (x + 1) - 2(x - 1) = -x + 3$$

Same as before, just derived differently.