ENGR 21:
Computer Engineering Fundamentals

Instructor: Emad Masroor

Lecture 21
Tuesday, November 18, 2025

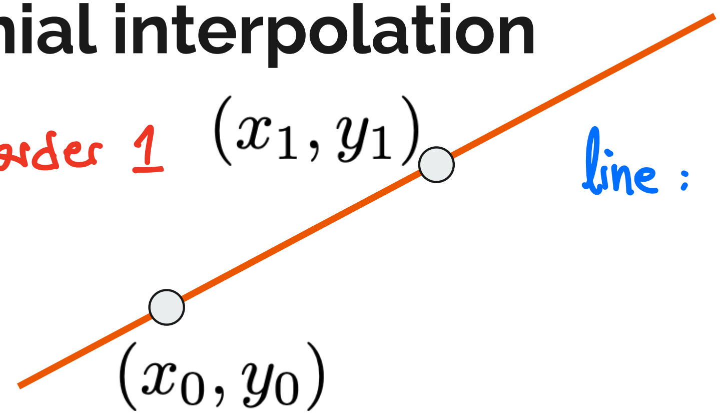
Polynomial Interpolation

Finding a straight line through 2 points is the simplest kind of polynomial interpolation

1. Write down the equation of a line

$$y = a_0 + a_1 x$$

order 1



line : polynomial of first order. $n = 1$

2. Plug in the data points

2 equations
2 points

$$\left.\begin{aligned} y_0 &= a_0 + a_1 x_0 \\ y_1 &= a_0 + a_1 x_1 \end{aligned}\right\} \to \text{a system of equations}$$

known: (x_i, y_i) for $i=0$ to $i=n$

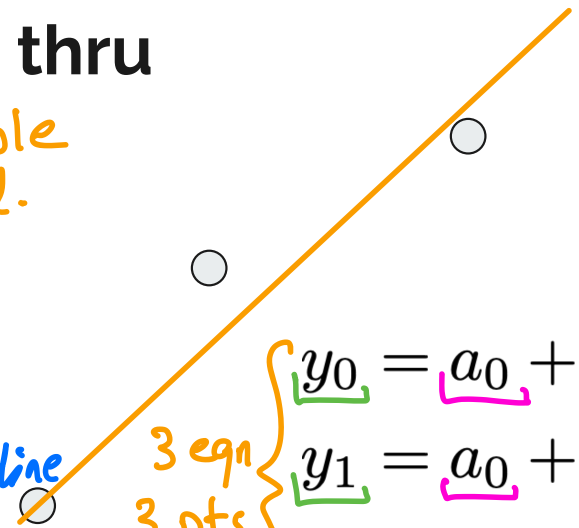
unknown: (a_0, a_1)

3. Solve simultaneous equations for the two unknowns

$$\begin{bmatrix} 1 & x_0 \\ 1 & x_1 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \end{bmatrix}$$

Finding a straight line that goes thru all three points

→ Not possible in general.



- Use 3 data points to attempt a straight line fit to three points.

→ Need a curve, not straight line

→ Let's use quadratic curve.

$$\text{3 eqn, 3 pts}\left\{\begin{aligned} y_0 &= a_0 + a_1 x_0 \\ y_1 &= a_0 + a_1 x_1 \\ y_2 &= a_0 + a_1 x_2 \end{aligned}\right.$$

$$\begin{bmatrix} 1 & x_0 \\ 1 & x_1 \\ 1 & x_2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \end{bmatrix}$$

Known

unknown

NOT close by ! exactly pass through

Finding a quadratic curve that goes thru all three points

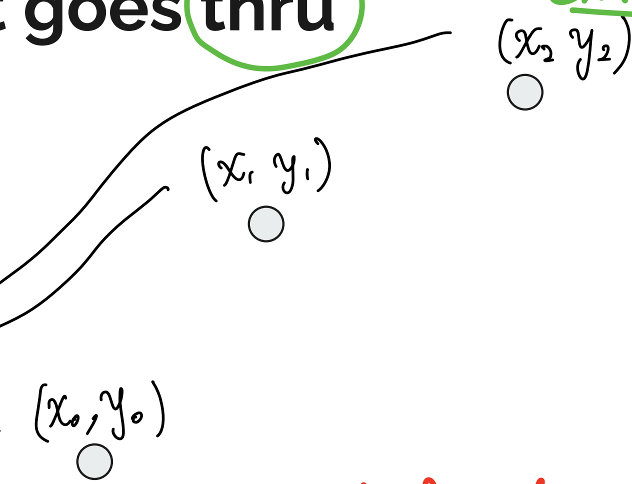
order $n=2$

1. Write down the equation of the curve

3 coefficients

$$y = a_0 + a_1 x + a_2 x^2$$

2. Plug in the data points



3 eqns.
3 pts.
3 unknowns

$$
\begin{aligned}
y_0 &= a_0 + a_1 x_0 + a_2 x_0^2 \\
y_1 &= a_0 + a_1 x_1 + a_2 x_1^2 \\
y_2 &= a_0 + a_1 x_2 + a_2 x_2^2
\end{aligned}
$$

A polynomial of order 2 can "interpolate between" 3 points

knowns
unknowns

3. Solve simultaneous equations for the three unknowns

unknowns

$$
\underbrace{\begin{bmatrix} 1 & x_0 & x_0^2 \\ 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \end{bmatrix}}_{\text{knowns}}
\begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix}
=
\underbrace{\begin{bmatrix} y_0 \\ y_1 \\ y_2 \end{bmatrix}}_{\text{knowns}}
$$

3x3 matrix
3x1 vector
3x1 vector

$$A\vec{x} = \vec{b}$$

matrix-vector equation can be solved with direct / iterative numerical methods.

Generalizing Polynomial Interpolation

You can (almost) always find a unique n-degree polynomial that goes through $\boxed{\text{n+1 points}}$ using a matrix of size (n+1) x (n+1)

known

$$
\begin{array}{c}
x_0^0 \\
x_1^0 \\
\\
x_n^0
\end{array}
\begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{bmatrix} \underbrace{\begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix}}_{\text{unknowns}} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ y_n \end{bmatrix}
$$

unknowns.
Solve for this

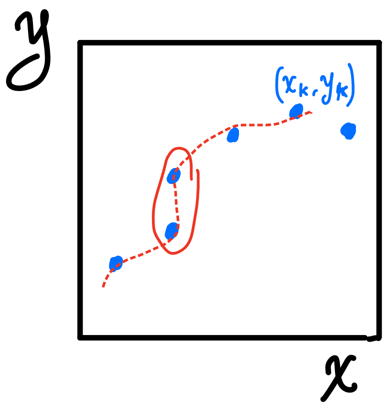
numpy.linalg.solve(A, b)

to build $y(x)$, given $a_0, a_1, \ldots, a_n$,
numpy.polyval

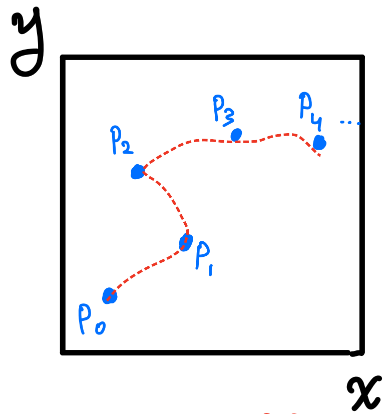
then you have a function,
$y(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + \ldots + a_nx^n$

and
$y(x_k) = y_k$ for all k from 0 to n

Potential Difficulties with Interpolation Matrices

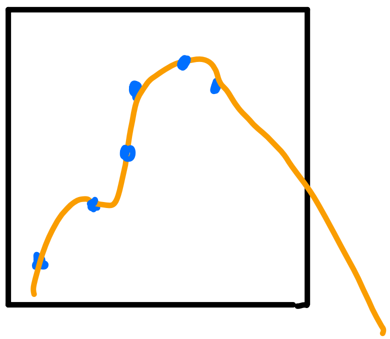


function would have to go vertical



Backtracking

outside range of data.



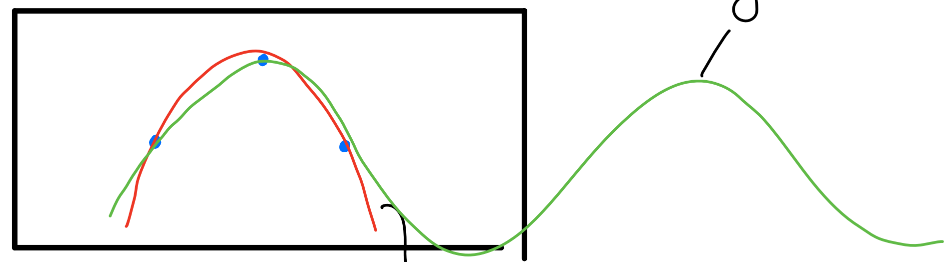
- only works in the region where you have data.
- Bad idea to extrapolate

[redacted]

→ Can also interpolate with other functions, not necessarily polynomials.

$\underbrace{\;}_{}$3 parameters

$y = a \sin bx + c$



quadratic polynomial

$y = \underbrace{a_0 + a_1 x + a_2 x^2}_{3 \text{ parameters}}$

Interpolating Matrices activity

a single line of code that solves $A\vec{x}=\vec{b}$

- Download coordinates from course website
- Write code that:
  - Generates the interpolating matrix for these points
  - Solves a linear system for the coefficients
- How would you use these coefficients to write the interpolating function?

```
from numpy.linalg import inv as invert
from numpy import dot
```

numpy.loadtxt

$\boxed{x = \mathrm{dot}(\mathrm{invert}(A), b)}$

x
1.0
2.0
2.5
3.0
3.5
4.5
6.0
9.0
9.5
10.0