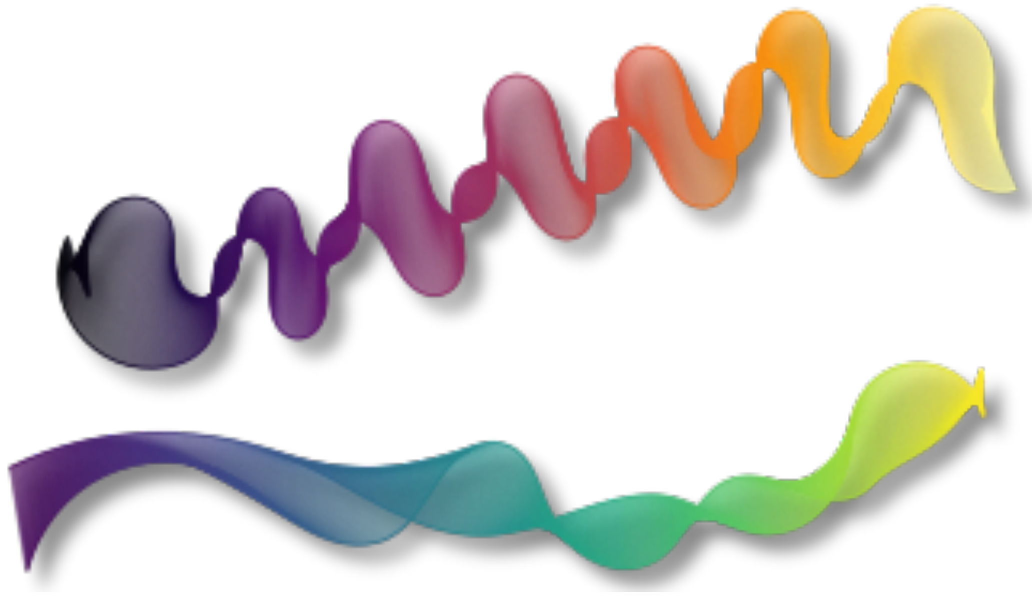
Prediction of Values with Polynomial

Guidance

Calculating interpolation using the Polynomial method



Depok, Indonesia

December 2025

Table of Contents

Polynomial Order

Matrix in Excel

Using Matrix

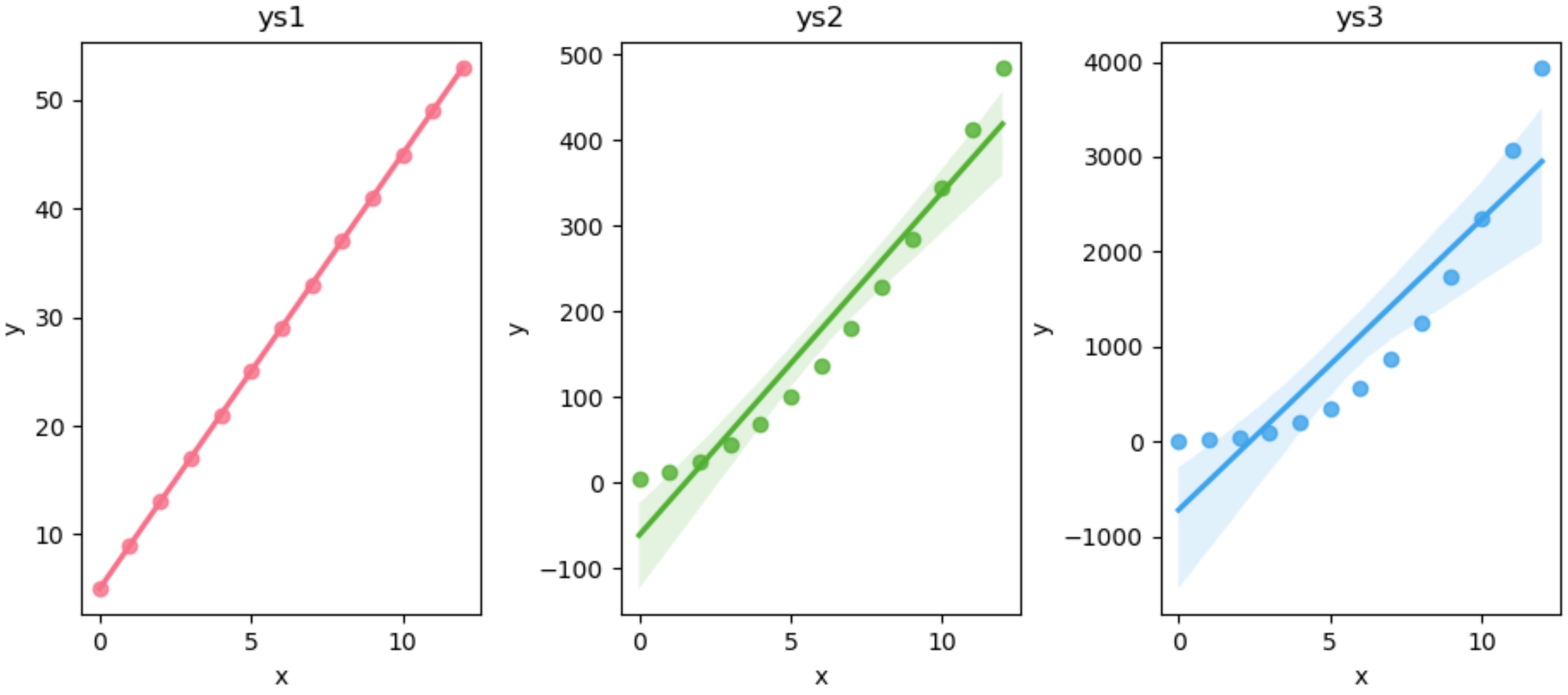
Charts in Matplotlib

Polynomial Order

Types of Interpolation

Interpolation: Linear

The purpose of interpolation is to predict values.
There are various types of interpolation, from polynomial to spline.



The simplest interpolation is linear interpolation.
Whatever the shape of the data, the result becomes a straight line.

Interpolation: Polinomial

Polynomial interpolation is chosen because it is easy to compute.

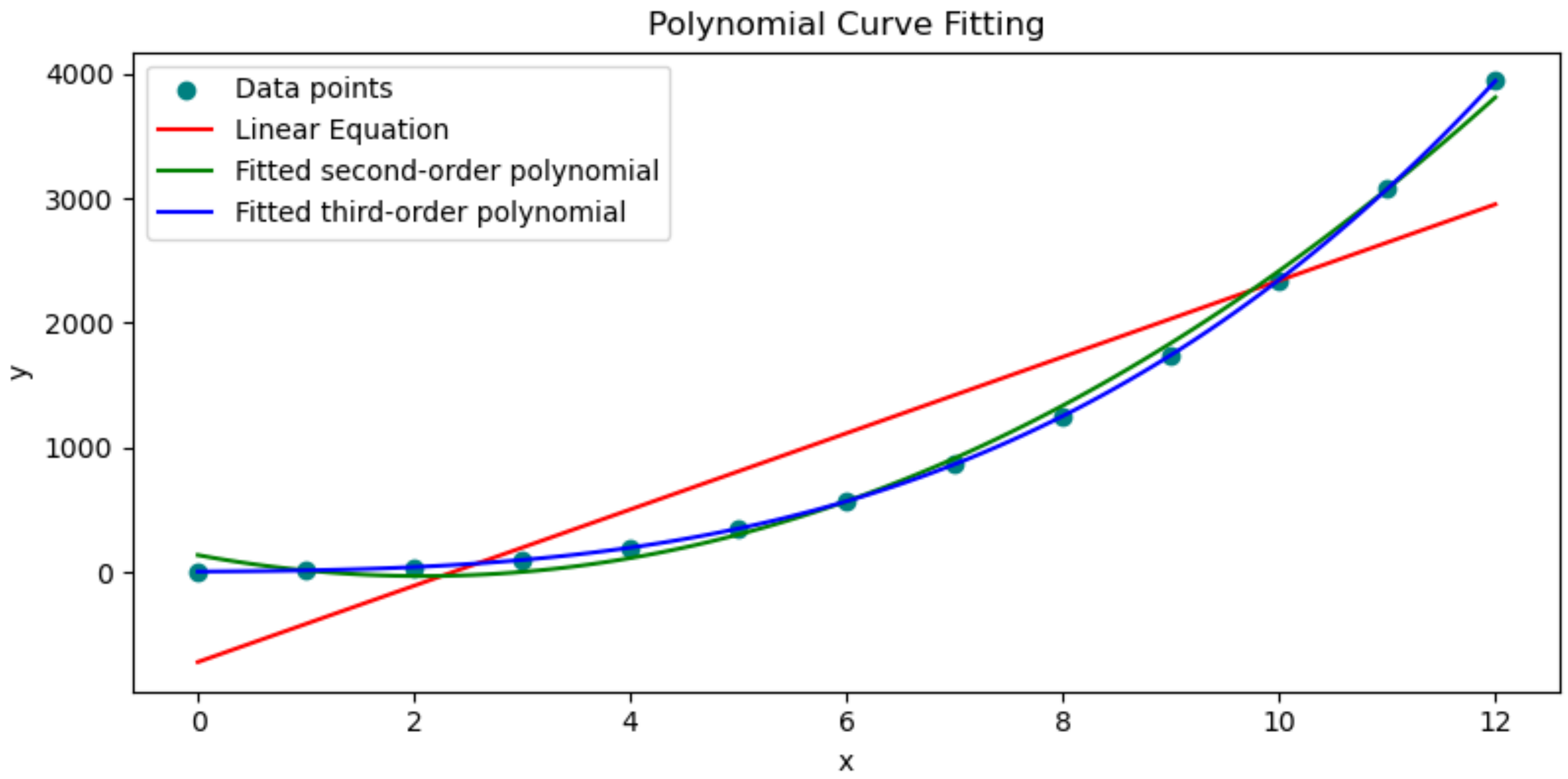
Polynomials have several different orders.
One, two, three, and so on.
The higher the order, the more coefficients it has.

$order = 1$	$y = a + bx$
$order = 2$	$y = a + bx + cx^2$
$order = 3$	$y = a + bx + cx^2 + dx^3$

The higher the polynomial order, the more accurate it becomes, but the more vulnerable it is to error.

Curve Fitting: Polynomial Order

Examples of polynomial orders: one (linear), two, and three.

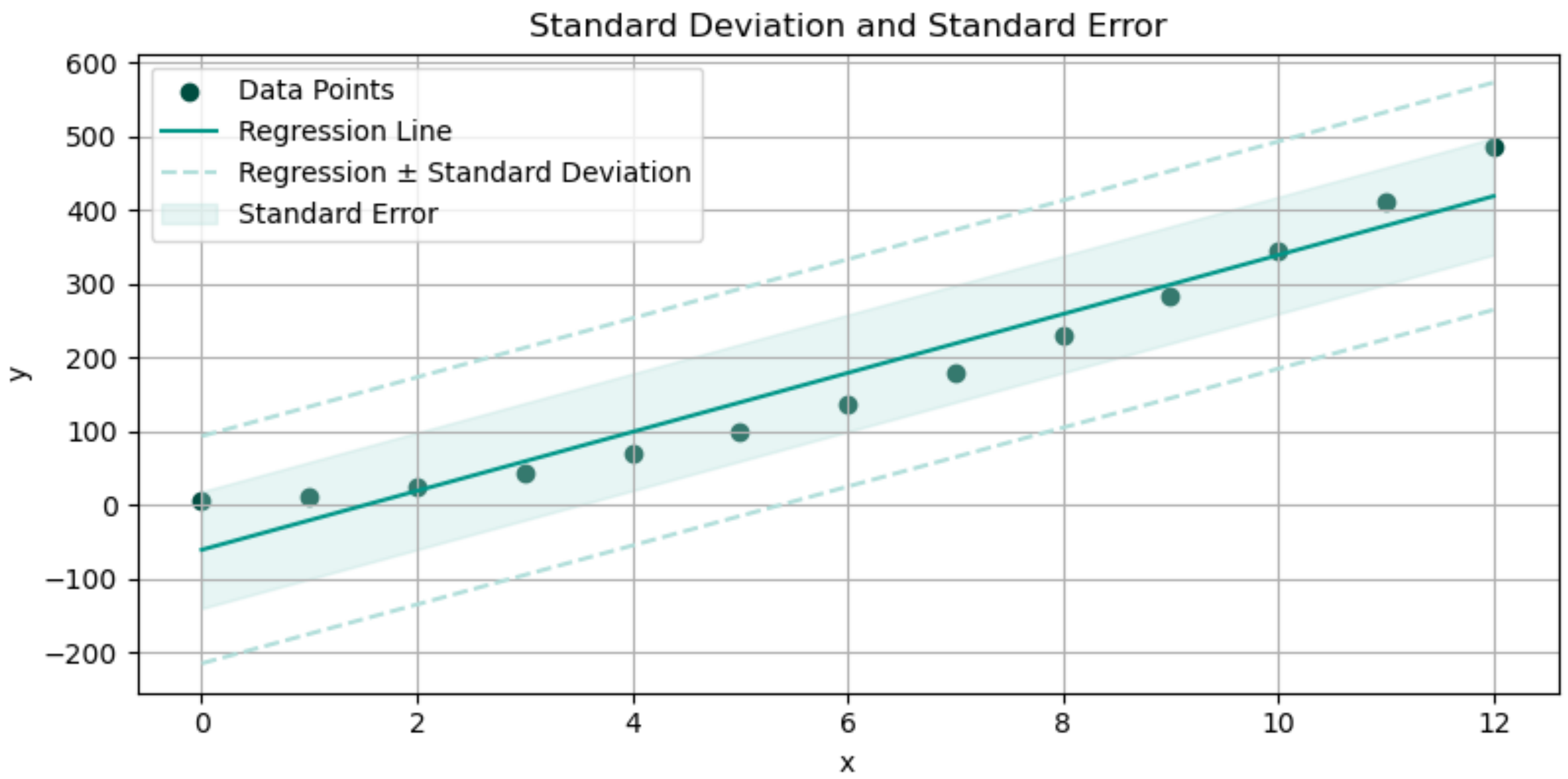


Usually, order two is already sufficient, with coefficients a, b, c.

$$y = a + bx + cx^2$$

Further Topic: Standard Deviation

This is an advanced topic, included in the regression guide.



It does not need to be discussed here yet,
but the material has been prepared in case it is needed later.

Using Matrix

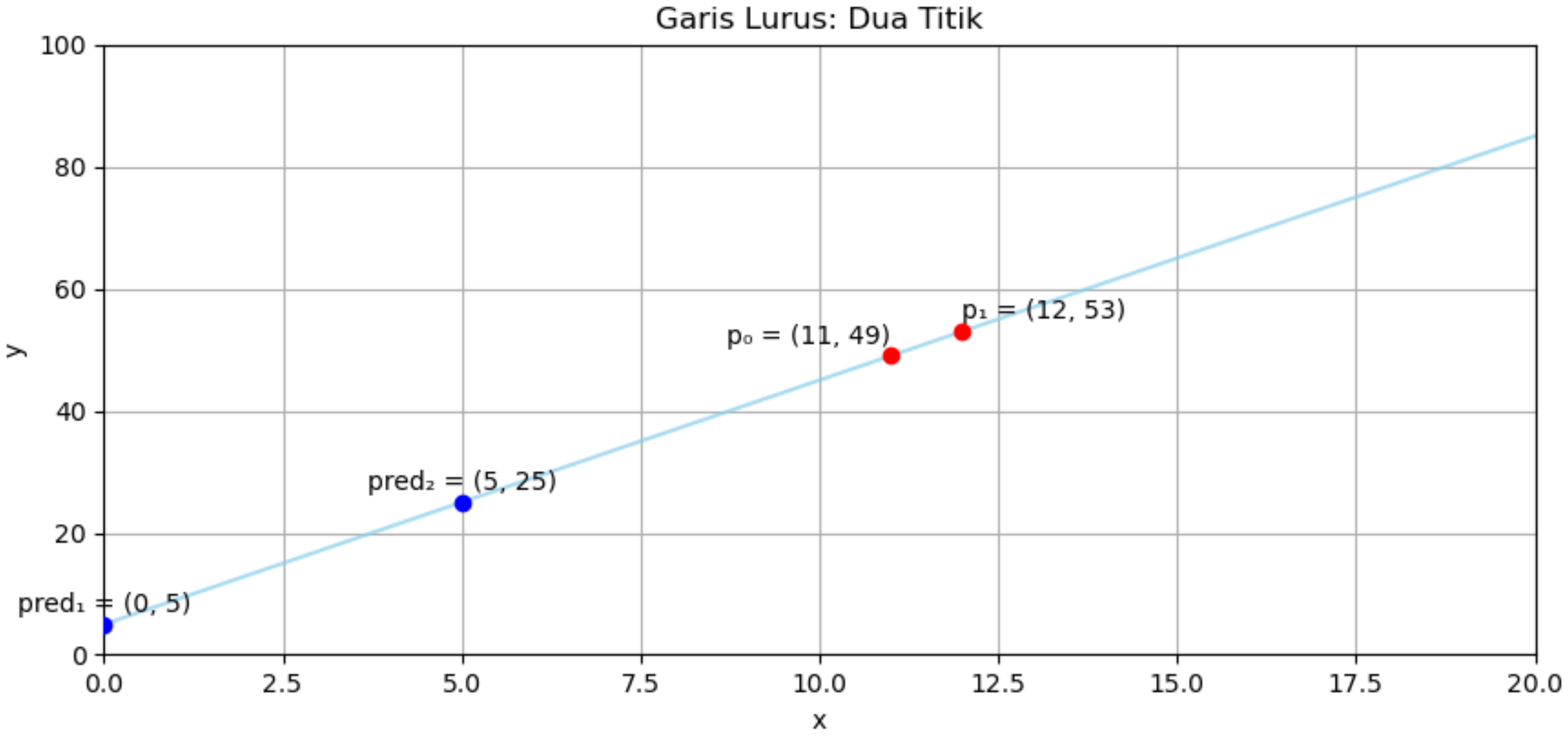
Analytical Method

Case Example: Two Points

From two points, predict other points!

Take an example of two points

- $p_0 = (11, 49)$
- $p_1 = (12, 53)$



We use the formula

$$y = mx + a$$

We will discuss it step by step,
so we understand if something strange (incorrect) appears.

Formula Method: Two Points

1st

Derive the formula

$$y = mx + a$$

$$m = \frac{\Delta y}{\Delta x}$$

$$m = \frac{y_2 - y_1}{x_2 - x_1}$$

$$a = y_1 - mx_1$$

2nd

Insert the two points

- $p_0 = (11, 49)$
- $p_1 = (12, 53)$

3rd

Insert into Excel

Slope Calculation

y = mx + a

$y - y_1 =$	$m(x - x_1)$
m =	$(y_2-y_1)/(x_2-x_1)$

x	known
11	49
12	53

m	a
4	5

4th

Apply the formula to predict

Equation Result

y = mx + a

coeff.	value
m	4,00000000
a	5,00000000

x	find	Formula
0	5	m.0+ a
5	25	m.5+ a
12	53	m.12+ a

Thus the result is obtained.

$$y = 5 + 4x$$

Matrix Method: Two Points

Review the two points again

$$p_0 = (x_0, y_0) \quad = (11, 49)$$
$$p_1 = (x_1, y_1) \quad = (12, 53)$$

They can be written in a (2x2) matrix as

$$\begin{bmatrix} y_0 \\ y_1 \end{bmatrix} = \begin{bmatrix} 1 & x_0 \\ 1 & x_1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} \quad \Rightarrow \quad \begin{bmatrix} 49 \\ 53 \end{bmatrix} = \begin{bmatrix} 1 & 11 \\ 1 & 12 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix}$$

Where:

$$A = \begin{bmatrix} 1 & 11 \\ 1 & 12 \end{bmatrix} \qquad B = \begin{bmatrix} 49 \\ 53 \end{bmatrix} \qquad C = \begin{bmatrix} a \\ b \end{bmatrix}$$

Coefficient C in the formula:

$$\begin{bmatrix} 1 & 11 \\ 1 & 12 \end{bmatrix} \times \mathbf{C} = \begin{bmatrix} 49 \\ 53 \end{bmatrix}$$

In matrix form, written again as

$$A \times \mathbf{C} = \mathbf{B}$$
$$\Rightarrow \qquad \mathbf{C} = A^{-1} \times \mathbf{B}$$

Matrix Method: Two Points

Using formula =minverse in Excel we get

$$A^{-1} = \begin{bmatrix} 12 & -11 \\ -1 & 1 \end{bmatrix}$$

Thus from the formula:

$$\mathbf{C} = A^{-1} \times \mathbf{B} = \begin{bmatrix} 5 \\ 4 \end{bmatrix}$$

From the obtained coefficients the equation becomes

$$y = 5 + 4x$$

Details of the solution:

$$\begin{bmatrix} 12 & -11 \\ -1 & 1 \end{bmatrix} \times \begin{bmatrix} 49 \\ 53 \end{bmatrix} = \begin{bmatrix} 12 \times 49 - 11 \times 53 \\ -1 \times 49 + 1 \times 53 \end{bmatrix}$$

$$= \begin{bmatrix} 588 - 583 \\ -49 + 53 \end{bmatrix}$$

$$= \begin{bmatrix} 5 \\ 4 \end{bmatrix}$$

Linear Interpolation Example

y = a + bx

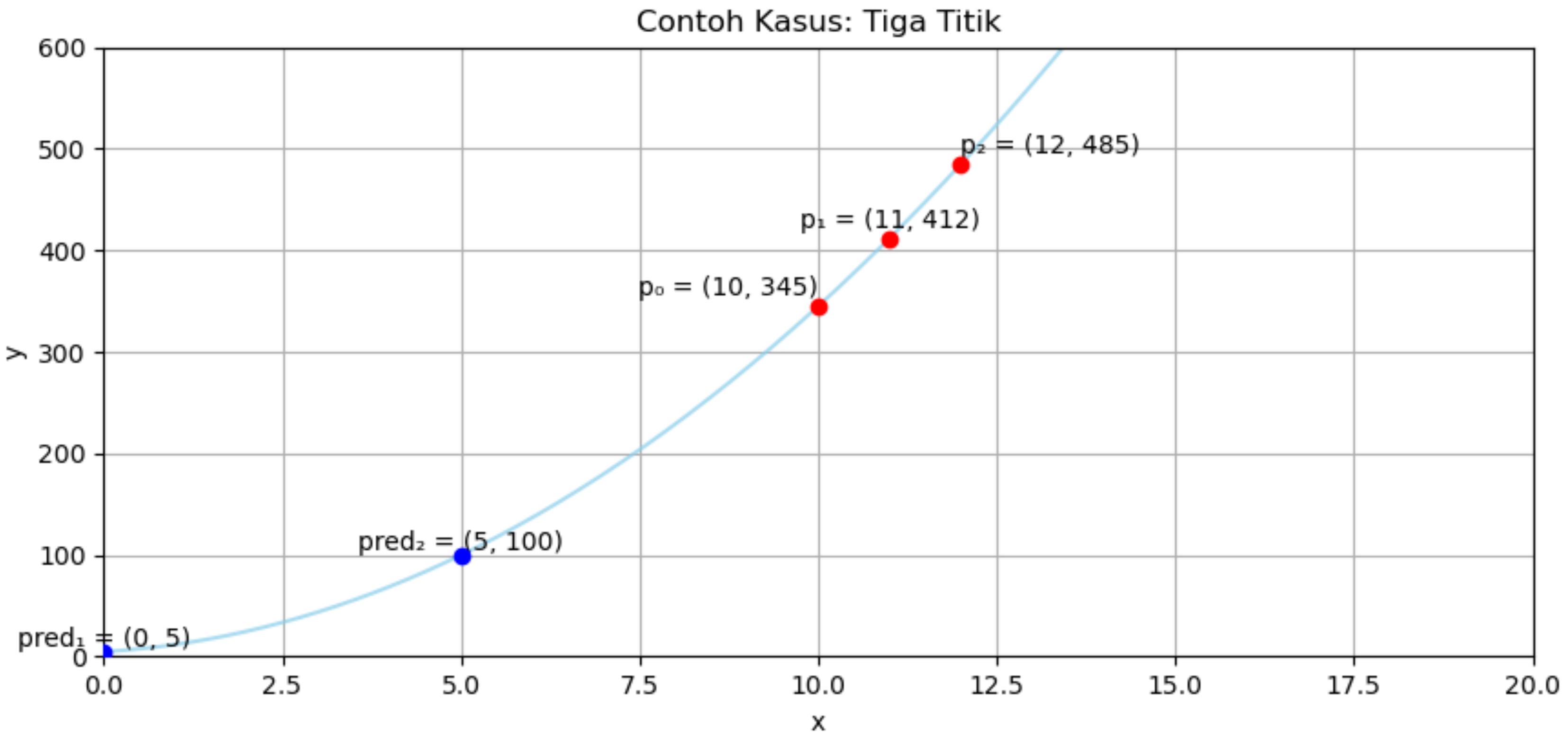
	A x C = B	**A =**	x1^0	x1^1					
≡	C = A⁻¹ x B		x2^0	x2^1					
x	known								
11	49	**A =**	1	11	**C =**	a	**B =**	49	
12	53		1	12		b		53	
C =	**A⁻¹ x B =**		5						
			4						

Case Example: Three Points

How to predict from three points?

Take an example of three points

- $p_0 = (10, 345)$
- $p_1 = (11, 412)$
- $p_2 = (12, 485)$



We use the quadratic formula

$$y = a + bx + cx^2$$

Rewrite into points

$$\begin{aligned} p_0 &= (x_0, y_0) &= (10, 345) \\ p_1 &= (x_1, y_1) &= (11, 412) \\ p_2 &= (x_2, y_2) &= (12, 485) \end{aligned}$$

Rewrite into matrix form (3x3)

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 & x_0 & {x_0}^2 \\ 1 & x_1 & {x_1}^2 \\ 1 & x_2 & {x_2}^2 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} \Rightarrow \begin{bmatrix} 345 \\ 412 \\ 485 \end{bmatrix} = \begin{bmatrix} 1 & 10 & 100 \\ 1 & 11 & 121 \\ 1 & 12 & 144 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

Matrix Method: Three Points

As usual:

$$A \times \mathbf{C} = \mathbf{B}$$
$$\Rightarrow \quad \mathbf{C} = A^{-1} \times \mathbf{B}$$

With the formula =minverse

$$A^{-1} = \begin{bmatrix} 66 & -20 & 55 \\ -11.5 & 22 & -10.5 \\ 0.5 & -1 & 0.5 \end{bmatrix}$$

The coefficients are:

$$\mathbf{C} = A^{-1}\mathbf{B} = \begin{bmatrix} 5 \\ 4 \\ 3 \end{bmatrix}$$

Example formula:

=MMULT(MINVERSE(E9:G11);K9:K11)

The equation is:

$$y = 5 + 4x + 3x^2$$

Second Order Interpolation Example

$y = a + bx + cx^2$

A x C = B

≡ C = A^{-1} x B

x	known
10	345
11	412
12	485

A = $\begin{vmatrix} 1 & 10 & 100 \\ 1 & 11 & 121 \\ 1 & 12 & 144 \end{vmatrix}$ **C =** $\begin{vmatrix} a \\ b \\ c \end{vmatrix}$ **B =** $\begin{vmatrix} 345 \\ 412 \\ 485 \end{vmatrix}$

C = **A^{-1} x B =** $\begin{vmatrix} 5 \\ 4 \\ 3 \end{vmatrix}$

Polynomial Result

$y = a + bx + cx^2$

coeff.	value
a	5,000
b	4,000
c	3,000

x	found	Formula
0	5	$a + b.0 + c.0^2$
5	100	$a + b.5 + c.5^2$
12	485	$a + b.12 + c.12^2$

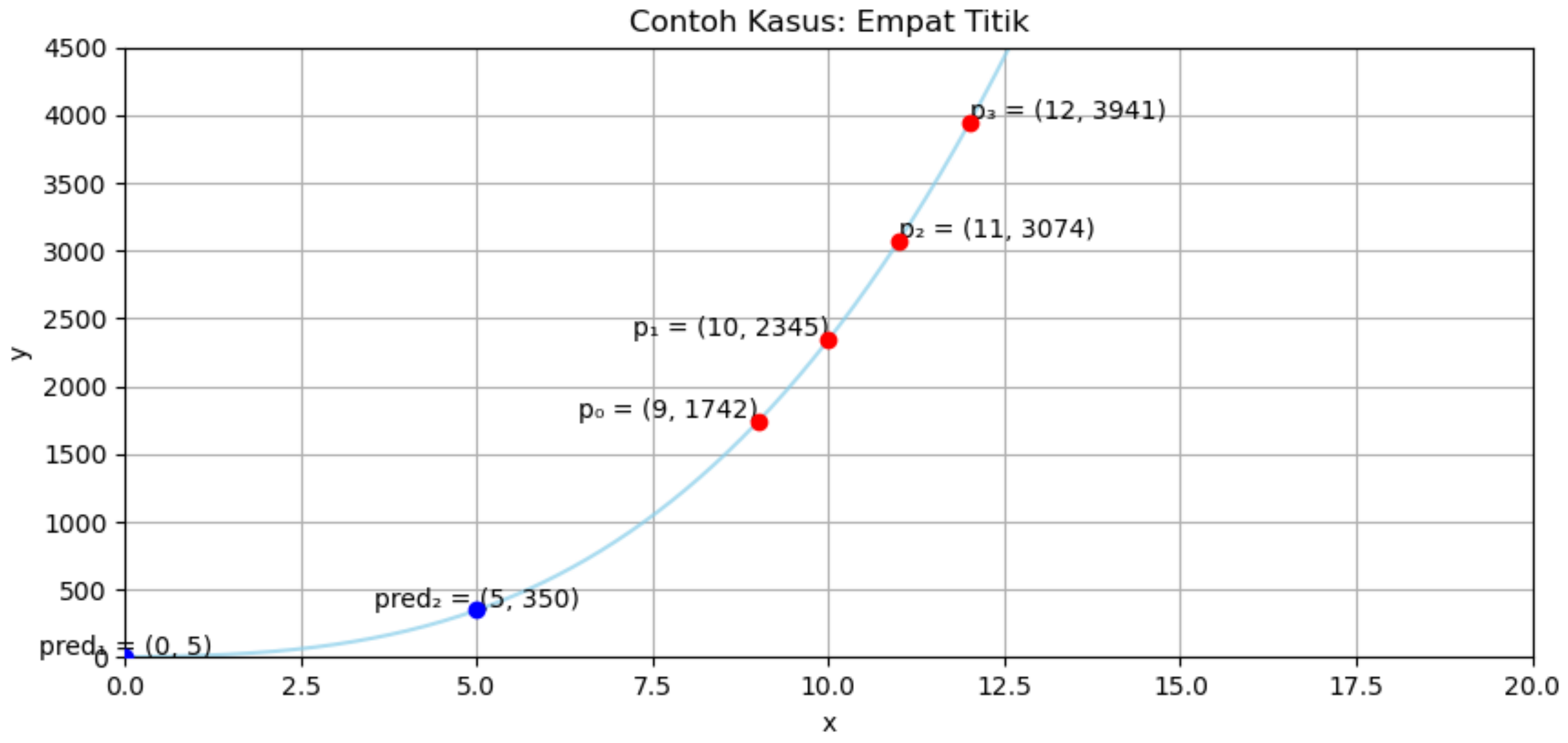
Case Example: Four Points

We use the cubic formula

$$y = a + bx + cx^2 + dx^3$$

Example of four points

$$\begin{aligned} p_0 &= (x_0, y_0) &&= (9, 1742) \\ p_1 &= (x_1, y_1) &&= (10, 2345) \\ p_2 &= (x_2, y_2) &&= (11, 3074) \\ p_3 &= (x_3, y_3) &&= (12, 3941) \end{aligned}$$



Rewrite into matrix form (4x4)

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 1 & x_0 & {x_0}^2 & {x_0}^3 \\ 1 & x_1 & {x_1}^2 & {x_0}^3 \\ 1 & x_2 & {x_2}^2 & {x_0}^3 \\ 1 & x_3 & {x_3}^2 & {x_0}^3 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} \Rightarrow \begin{bmatrix} 1742 \\ 2345 \\ 3074 \\ 3941 \end{bmatrix} = \begin{bmatrix} 1 & 9 & 81 & 729 \\ 1 & 10 & 100 & 1000 \\ 1 & 11 & 121 & 1331 \\ 1 & 12 & 144 & 1728 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}$$

Matrix Method: Four Points

As usual:

$$A \times \mathbf{C} = \mathbf{B}$$

$$\Rightarrow \quad \mathbf{C} = A^{-1} \times \mathbf{B}$$

The coefficients are:

$$\mathbf{C} = A^{-1}\mathbf{B} = \begin{bmatrix} 5 \\ 4 \\ 3 \\ 2 \end{bmatrix}$$

Then the coefficients are:

$$y = 5 + 4x + 3x^2 + 2x^3$$

Then the coefficients are:

=MMULT(MINVERSE(E9:G11);K9:K11)

Third Order Interpolation Example

$y = a + bx + cx^2 + dx^3$

$A \times C = B$

$\equiv \quad C = A^{-1} \times B$

x	known
9	1.742
10	2.345
11	3.074
12	3.941

A =

$$\begin{vmatrix} 1 & 9 & 81 & 729 \\ 1 & 10 & 100 & 1.000 \\ 1 & 11 & 121 & 1.331 \\ 1 & 12 & 144 & 1.728 \end{vmatrix}$$

C =

$$\begin{vmatrix} a \\ b \\ c \\ d \end{vmatrix}$$

B =

$$\begin{vmatrix} 1.742 \\ 2.345 \\ 3.074 \\ 3.941 \end{vmatrix}$$

C = **$A^{-1} \times B$ =**

$$\begin{vmatrix} 5 \\ 4 \\ 3 \\ 2 \end{vmatrix}$$

Polynomial Result

$y = a + bx + cx^2 + dx^3$

coeff.	value
a	5,000
b	4,000
c	3,000
d	2,000

x	found	Formula
0	5	$a + b.0 + c.0^2 + d.0^3$
5	350	$a + b.5 + c.5^2 + d.5^3$
12	3.941	$a + b.12 + c.12^2 + d.12^3$

Generalization of Calculation:
Polynomial Cheatsheet

Data Series (x, y) :

x_i (observed) $= [\ldots]$,

y_i (observed) $= [\ldots]$

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \cdots + \beta_k x^k$$

$(explicit form)$

$$\begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^k \\ 1 & x_2 & x_2^2 & \cdots & x_2^k \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^k \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_k \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

$(matrix form)$

1st $y = \beta_0 + \beta_1 x$

2nd $y = \beta_0 + \beta_1 x + \beta_2 x^2$

3rd $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta\ x$

(polynomial degree)

$$X\beta = \mathbf{y}$$

$$\Rightarrow \quad X^\top X\beta = X^\top \mathbf{y}$$

$$\Rightarrow \quad \beta = (X^\top X)^{-1} X^\top \mathbf{y}$$

$(solution)$

Linear	Quadratic	Cubic
$\mathbf{X} = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix}, \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$	$\mathbf{X} = \begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 \end{bmatrix}, \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$	$\mathbf{X} = \begin{bmatrix} 1 & x_1 & x_1^2 & x_1^3 \\ 1 & x_2 & x_2^2 & x_2^3 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 & x_n^3 \end{bmatrix}, \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$
$(\mathbf{X}^T\mathbf{X}) \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} = \mathbf{X}^T\mathbf{y}$	$(\mathbf{X}^T\mathbf{X}) \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \end{bmatrix} = \mathbf{X}^T\mathbf{y}$	$(\mathbf{X}^T\mathbf{X}) \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta 3 \end{bmatrix} = \mathbf{X}^T\mathbf{y}$

$\hat{\boldsymbol{\beta}} = (X^\top X)^{-1} X^\top \mathbf{y}$ $\longrightarrow$ $\hat{y}_i = \mathrm{Model}(x_i)$ $\longrightarrow$ $\hat{y}\mathbf{i}$(predicted$\Rightarrow= [\ldots]$

(use matrix operations to find coefficients)

Vandermonde-based least squares

(Ref:Wikipedia,2023)

Matrix in Excel

Application of the Analytical Method

Example Data: From equation to matrix

Let us create example interpolation data,
from an equation with known coefficients,
simply with 13 points (x, y), then tabulate them into Excel.

$$y = 5 + 4x + 3x^2 + 2x^3$$

$$x_i = [0, 1, 2, \ldots, 10, 11, 12]$$

$$y_i = a + bx_i$$

$$\begin{bmatrix} 1 & x_0 & {x_0}^2 & {x_0}^3 \\ 1 & x_1 & {x_1}^2 & {x_1}^3 \\ 1 & x_2 & {x_2}^2 & {x_2}^3 \\ 1 & x_3 & {x_3}^2 & {x_3}^3 \\ 1 & x_4 & {x_4}^2 & {x_4}^3 \\ 1 & x_5 & {x_5}^2 & {x_5}^3 \\ 1 & x_6 & {x_6}^2 & {x_6}^3 \\ 1 & x_7 & {x_7}^2 & {x_7}^3 \\ 1 & x_8 & {x_8}^2 & {x_8}^3 \\ 1 & x_9 & {x_9}^2 & {x_9}^3 \\ 1 & x_{10} & {x_{10}}^2 & {x_{10}}^3 \\ 1 & x_{11} & {x_{11}}^2 & {x_{11}}^3 \\ 1 & x_{12} & {x_{12}}^2 & {x_{12}}^3 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \\ y_8 \\ y_9 \\ y_{10} \\ y_{11} \\ y_{12} \end{bmatrix}$$

Matrix

$y = a + bx + cx^2 + dx^3$

x	desired	P(x)	x^0	x^1	x^2	x^3	coeff.	value
0	5	P(0)	1	0	0	0	5	5
1	14	P(1)	1	1	1	1	4	14
2	41	P(2)	1	2	4	8	3	41
3	98	P(3)	1	3	9	27	2	98
4	197	P(4)	1	4	16	64		197
5	350	P(5)	1	5	25	125		350
6	569	P(6)	1	6	36	216		569
7	866	P(7)	1	7	49	343		866
8	1,253	P(8)	1	8	64	512		1253
9	1,742	P(9)	1	9	81	729		1742
10	2,345	P(10)	1	10	100	1,000		2345
11	3,074	P(11)	1	11	121	1,331		3074
12	3,941	P(12)	1	12	144	1,728		3941

We will reverse from matrix to the formula coefficients.

Interpolation Data: Matrix n x m

When performing interpolation with many data points,
the matrix used is not a square matrix (n x n),
but a rectangular matrix (n x m), for example here (13 x 4).

In practice we will use Excel or a Python Script.

Let us take example data with 13 rows, then insert into Excel.

Range A = E9:H21
Range B = L9:L21

Inverse

$y = a + bx + cx^2 + dx^3$

$A \times C = B$

$\equiv \quad C = A^{-1} \times B$

$$A \times \mathbf{C} = \mathbf{B}$$

$$\Rightarrow \quad A^T \times A \times \mathbf{C} = A^T \times \mathbf{B}$$

$$\Rightarrow \quad \mathbf{C} = (A^T \times A)^{-1} \times (A^T \times \mathbf{B})$$

x	known	A =					C =		B =	
0	5		1	0	0	0		a		5
1	14		1	1	1	1		b		14
2	41		1	2	4	8		c		41
3	98		1	3	9	27		d		98
4	197		1	4	16	64				197
5	350		1	5	25	125				350
6	569		1	6	36	216				569
7	866		1	7	49	343				866
8	1,253		1	8	64	512				1,253
9	1,742		1	9	81	729				1,742
10	2,345		1	10	100	1,000				2,345
11	3,074		1	11	121	1,331				3,074
12	3,941		1	12	144	1,728				3,941

Inverse: n x m Matrix

$$
\begin{aligned}
& AC = \mathbf{B} \\
\Rightarrow \quad & A^T AC = A^T\mathbf{B} \\
\Rightarrow \quad & \mathbf{C} = (A^T A)^{-1} A^T \mathbf{B}
\end{aligned}
$$

Because inverse applies only to square matrix (n x n), the basic equation must be modified slightly, namely by performing a transpose first.

Inverse

$y = a + bx + cx^2 + dx^3$

$$A^t \times A = \begin{vmatrix} 13 & 78 & 650 & 6084 \\ 78 & 650 & 6084 & 60710 \\ 650 & 6084 & 60710 & 630708 \\ 6084 & 60710 & 630708 & 6735950 \end{vmatrix}$$

$$(A^t \times A)^{-1} = \begin{vmatrix} 0{,}728022 & -0{,}43086 & 0{,}068681 & -0{,}00321 \\ -0{,}43086 & 0{,}41197 & -0{,}07855 & 0{,}004031 \\ 0{,}068681 & -0{,}07855 & 0{,}016234 & -0{,}00087 \\ -0{,}00321 & 0{,}004031 & -0{,}00087 & 4{,}86E\text{-}05 \end{vmatrix}$$

$$A^t \times B = \begin{vmatrix} 14495 \\ 142662 \\ 1471132 \\ 15637284 \end{vmatrix}$$

$$C = \quad (A^t \times A)^{-1} \times A^t \times B = \begin{vmatrix} 5 \\ 4 \\ 3 \\ 2 \end{vmatrix}$$

Formula in Excel.	Range in Worksheet
A^t {=TRANSPOSE(E9:H21)}	A^t = E27:Q30
A^t x A {=MMULT(E27:Q30,E9:H21)}	A^t x A = E32:H35
$(A^t \text{ x } A)^{-1}$ {=MINVERSE(E32:H35)}	$(A^t \text{ x } A)^{-1}$ = E37:H40
A^t x B {=MMULT(E27:Q30,L9:L21)}	A^t x B = E42:E45
$(A^t \text{ x } A)^{-1}$ x A^t x B C {=MMULT(E37:H40,E42:E45)}	C = E47:E50

$$y = 5 + 4x + 3x^2 + 2x^3$$

Combined: Gram Matrix

To simplify comparison, we can combine
all calculations in one sheet, using the Gram Matrix Gram Matrix ($A^t.A$).

Linear $y = a + bx$

Gram Matrix ($A^t.A$)	
13	78
78	650

Inverse Matrix ($A^t.A)^{-1}$	
0.2747	-0.0330
-0.0330	0.0055

$A^t.B$
14,495
142,662

	Coefficients
a	-721
b	306

Quadratic $y = a + bx + cx^2$

Gram Matrix ($A^t.A$)		
13	78	650
78	650	6,084
650	6,084	60,710

Inverse Gram Matrix ($A^t.A)^{-1}$		
0.5165	-0.1648	0.0110
-0.1648	0.0774	-0.0060
0.0110	-0.0060	0.0005

$A^t.B$
14,495
142,662
1,471,132

	Coefficients
a	137
b	-162
c	39

Cubic $y = a + bx + cx^2 + dx^3$

Gram Matrix ($A^t.A$)			
13	78	650	6,084
78	650	6,084	60,710
650	6,084	60,710	630,708
6,084	60,710	630,708	6,735,950

Inverse Gram Matrix ($A^t.A)^{-1}$			
0.7280	-0.4309	0.0687	-0.0032
-0.4309	0.4120	-0.0785	0.0040
0.0687	-0.0785	0.0162	-0.0009
-0.0032	0.0040	-0.0009	0.0000

($A^t.B$)
14,495
142,662
1,471,132
15,637,284

	Coefficients
a	5
b	4
c	3
d	2

Because everything is inside one worksheet,
we can directly compare easily.

We will encounter Gram Matrix ($A^t.A$) again,
when calculating polynomial regression (not linear regression).

Example Formula for a Straight Line

Range: B37:C49
x: B37:B49
y: C37:C49

A
=CHOOSE({1,2}, 1, B37:B49)

B: Range: C37:C49

A^t
=TRANSPOSE(A)
=TRANSPOSE(CHOOSE({1,2}, 1, B37:B49))

($A^t.A$)
Range: B12:C13
=MMULT(TRANSPOSE(A), A)
=MMULT(TRANSPOSE(CHOOSE({1,2}, 1, B37:B49)), CHOOSE({1,2}, 1, B37:B49))

($A^t.A)^{-1}$
Range: B18:C19
=MINVERSE(B12:C13)

($A^t.B$)
Range: B24:B25
=MMULT(TRANSPOSE(A), B)
=MMULT(TRANSPOSE(CHOOSE({1,2}, 1, B37:B49)), C37:C49)

C: Coefficients ($A^t.A)^{-1}.(A^t.B)$
=MMULT(B18:C19, B24:B25)

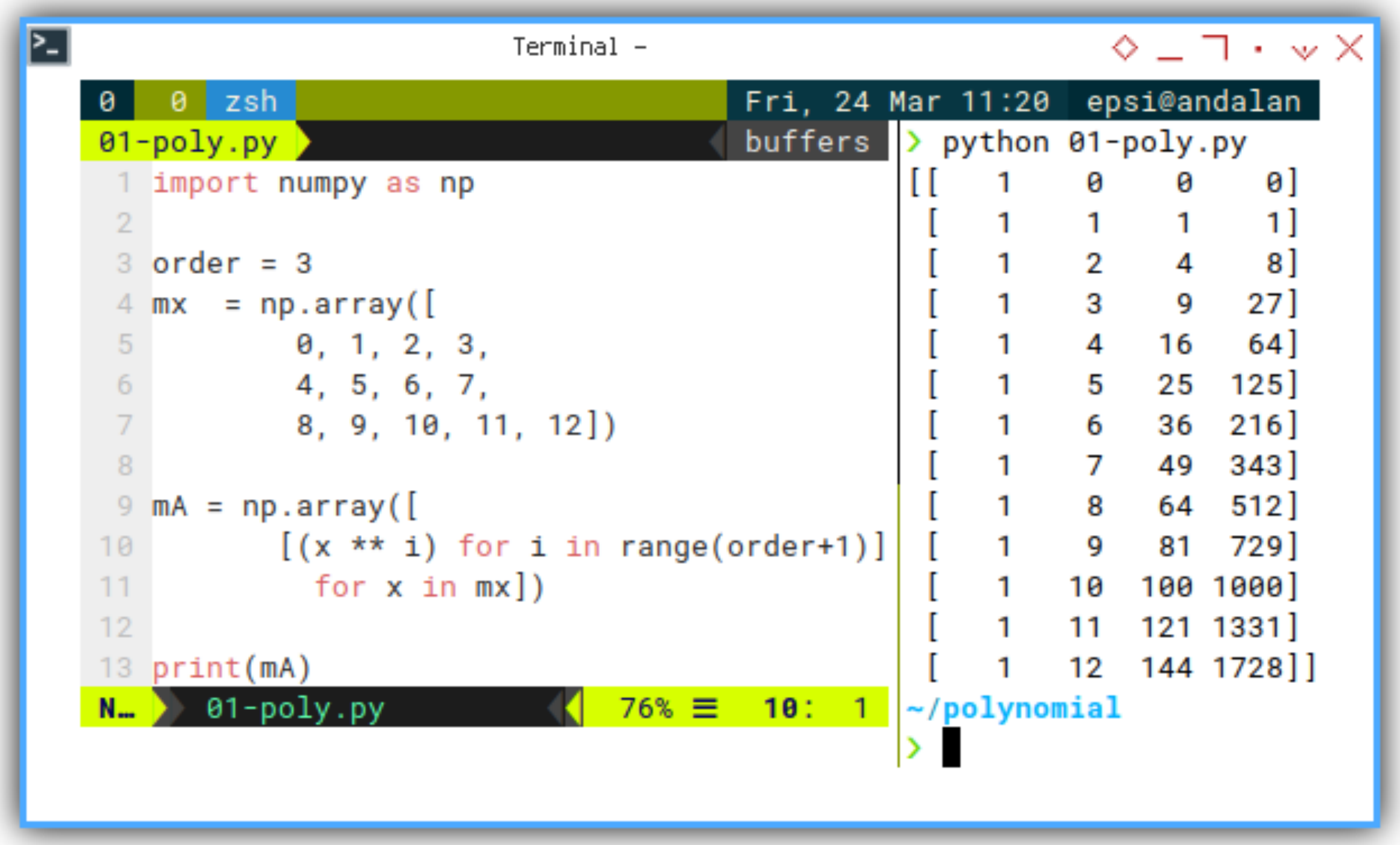
Chart in Matplotlib

Curve Fitting in Python

Python: List Comprehension

We will use Python, which is easy to use.
With List Comprehension, matrix are very easy to create.

```
import numpy as np

order = 3
mx   = np.array([
        0, 1, 2, 3,
        4, 5, 6, 7,
        8, 9, 10, 11, 12])

mA = np.array([
       [(x ** i) for i in range(order+1)]
         for x in mx])

print(mA)
```



In Python there is a Vandermonde function:
mV = np.vander(mx, 4)

However, the order of the coefficients needs to be reversed:

$$y = ax^3 + bx^2 + cx + d$$

Python: Matrix Operations

With the matrix available, we can multiply easily.

```python
import numpy as np

order = 3
mx  = np.array([
        0, 1, 2, 3,
        4, 5, 6, 7,
        8, 9, 10, 11, 12])
mB  = np.array([
         5,   14,   41,   98,
       197,  350,  569,  866,
      1253, 1742, 2345, 3074, 3941])

mA = np.array([
       [(x ** i) for i in range(order+1)]
         for x in mx])

```

```python
mAt   = np.transpose(mA)
mAt_A = np.matmul(mAt, mA)
mAt_B = np.matmul(mAt, mB)

# First Method
mAt_A_i = np.linalg.inv(mAt_A)
mC      = np.matmul(mAt_A_i, mAt_B)
print("Coefficients (a, b, c, d):", mC)

# Second Method
mC      = np.linalg.solve(mAt_A, mAt_B)
print("Coefficients (a, b, c, d):", mC)
```

```
❯ python 02-poly.py
Coefficients (a, b, c, d): [5. 4. 3. 2.]
Coefficients (a, b, c, d): [5. 4. 3. 2.]
```

We can utilize the @ operator to multiply matrices.

mC = mAt_A_i @ mAt_B

Python: Matplotlib

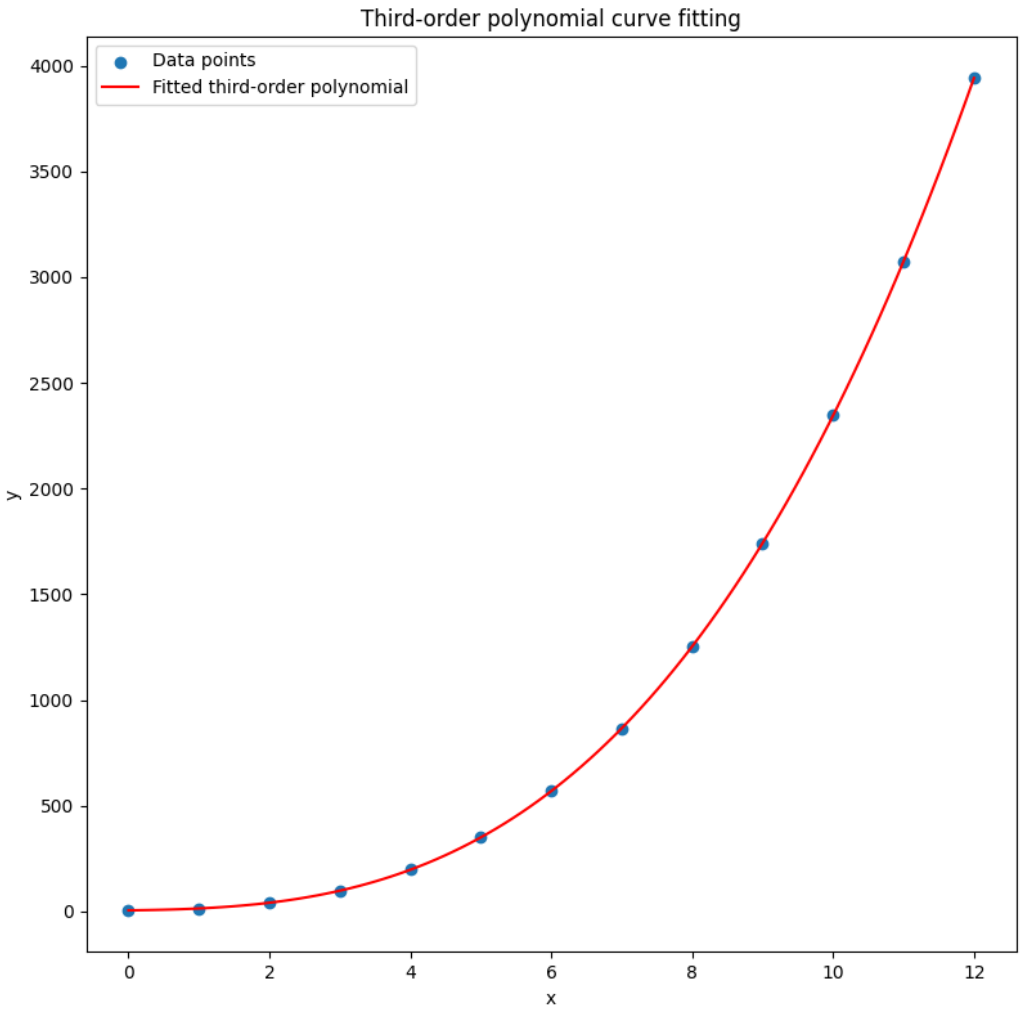
04-poly.py

```python
# Calculated Matrix Variable
mA    = np.flip(np.vander(mx, 4), axis=1)
mAt   = np.transpose(mA)
mAt_A = mAt @ mA
mAt_B = mAt @ mB
mC    = np.linalg.solve(mAt_A, mAt_B)
[a, b, c, d] = mC
print("Coefficients (a, b, c, d):", mC)
```

03-poly.py

```python
# Draw Plot
[a, b, c, d] = mC

x_plot = np.linspace(min(mx), max(mx), 100)
y_plot = a + b * x_plot + \
         c * x_plot**2 + d * x_plot**3

plt.scatter(mx, mB, label='Data points')
plt.plot(x_plot, y_plot, color='red',
  label='Fitted third-order polynomial')

plt.legend()
plt.xlabel('x')
plt.ylabel('y')
plt.title(
  'Third-order polynomial curve fitting')
```

Now we can pour
the results into chart form.



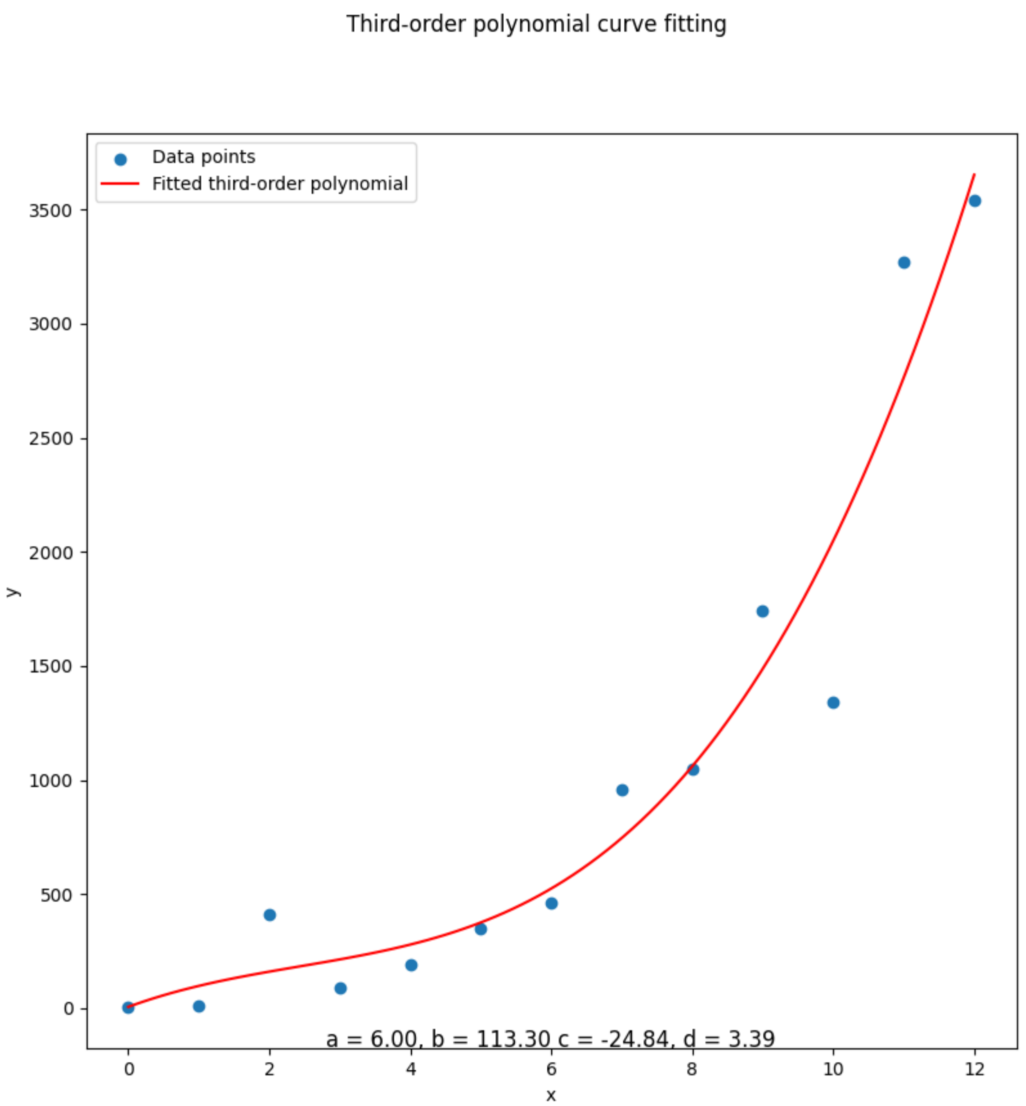
Python: Curve Fitting

```python
order = 3
mx  = np.array([
        0, 1, 2, 3,
        4, 5, 6, 7,
        8, 9, 10, 11, 12])
mB  = np.array([
        5,   10,   410,   90,
       190,  350,  460,  960,
     1050, 1740, 1340, 3270, 3540])
```

```python
# Draw Plot
[a, b, c, d] = mC

+--- 11 lines: x_plot = np.linspace(min(mx), max(m
plt.suptitle(
  'Third-order polynomial curve fitting')

subfmt = "a = %.2f, b = %.2f c = %.2f, d = %.2f"
plt.title(subfmt % (a, b, c, d), y=-0.01)

plt.show()
```

Let us test for different data.

And also display the equation coefficients.



Data Source: Two CSV Examples

We prepare two CSV files,
the first from a formula, the second with more random data.

```
x,y
0,5
1,14
2,41
3,98
4,197
5,350
6,569
7,866
8,1253
9,1742
10,2345
11,3074
12,3941
```

```
x,y
0,5
1,10
2,410
3,90
4,190
5,350
6,460
7,960
8,1050
9,1740
10,1340
11,3270
12,3540
```

Python Script: Polyfit dan Matplotlib

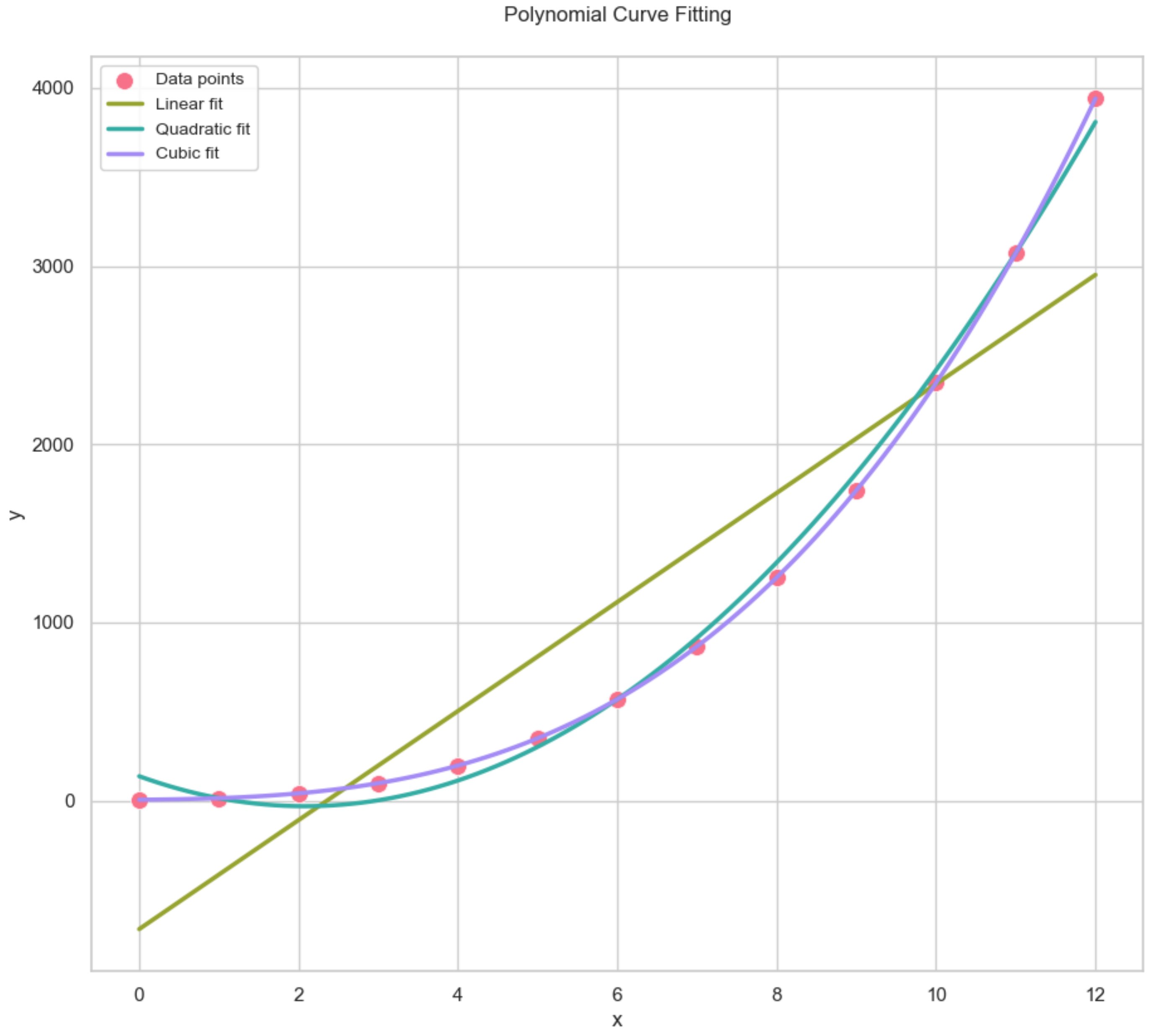
Script so short that we can self learnig.

```python
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

from numpy.polynomial import Polynomial
from typing import List

class CurveFitting:
    def __init__(self, xs, ys : List[int]) -> None:
        # Given data
        self.xs = np.array(xs)
        self.ys = np.array(ys)

        # Display
        self.coeff_text = {
            1: '(a, b)', 2: '(a, b, c)', 3: '(a, b, c, d)'}
        self.order_text = {
            1: 'Linear',  2: 'Quadratic ', 3: 'Cubic'}

        # Seaborn styling setup
        sns.set_theme(style="whitegrid")
        self.colors = sns.color_palette("husl", 4)

    def print_props(self, order) -> np.ndarray:
        # Perform regression using polyfit,
        poly = Polynomial.fit(self.xs, self.ys, deg=order)

        # Convert to standard form and get coefficients
        mC = poly.convert().coef

        print(f'Using Polynomial.fit : {self.order_text[order]}')
        print(f'Coefficients  : {self.coeff_text[order]}:'
            + f'\n\t{mC}\n')

    def calc_plot_all(self) -> None:
        self.x_plot = xp = np.linspace(
            min(self.xs), max(self.xs), 100)

        # Calculate coefficients directly
        self.y1_plot = Polynomial.fit(self.xs, self.ys, deg=1)(xp)
        self.y2_plot = Polynomial.fit(self.xs, self.ys, deg=2)(xp)
        self.y3_plot = Polynomial.fit(self.xs, self.ys, deg=3)(xp)

    def draw_plot(self) -> None:
        plt.figure(figsize=(10, 6))

        # Scatter plot with Seaborn color
        sns.scatterplot(
            x=self.xs, y=self.ys, color=self.colors[0],
            s=100, label='Data points', edgecolor='w', linewidth=0.5)

        # Polynomial curves with Seaborn colors
        plt.plot(self.x_plot, self.y1_plot, color=self.colors[1],
                linewidth=2.5, label='Linear fit')
        plt.plot(self.x_plot, self.y2_plot, color=self.colors[2],
                linewidth=2.5, label='Quadratic fit')
        plt.plot(self.x_plot, self.y3_plot, color=self.colors[3],
                linewidth=2.5, label='Cubic fit')

        # Styling
        plt.title('Polynomial Curve Fitting', pad=20)
        plt.xlabel('x', fontsize=12)
        plt.ylabel('y', fontsize=12)

        # Legend and grid
        plt.legend(fontsize=10, framealpha=0.9)

        plt.tight_layout()
        plt.show()

    def process(self) -> None:
        self.calc_plot_all()
        self.draw_plot()

        for order in [1, 2, 3]:
            self.print_props(order)

def main() -> int:
    # Getting Matrix Values
    mCSV = np.genfromtxt("polynomial.csv",
      skip_header=1, delimiter=",", dtype=float)
    mCSVt   = np.transpose(mCSV)

    example = CurveFitting(mCSVt[0], mCSVt[1])
    example.process()

    return 0

if __name__ == "__main__":
    raise SystemExit(main())
```

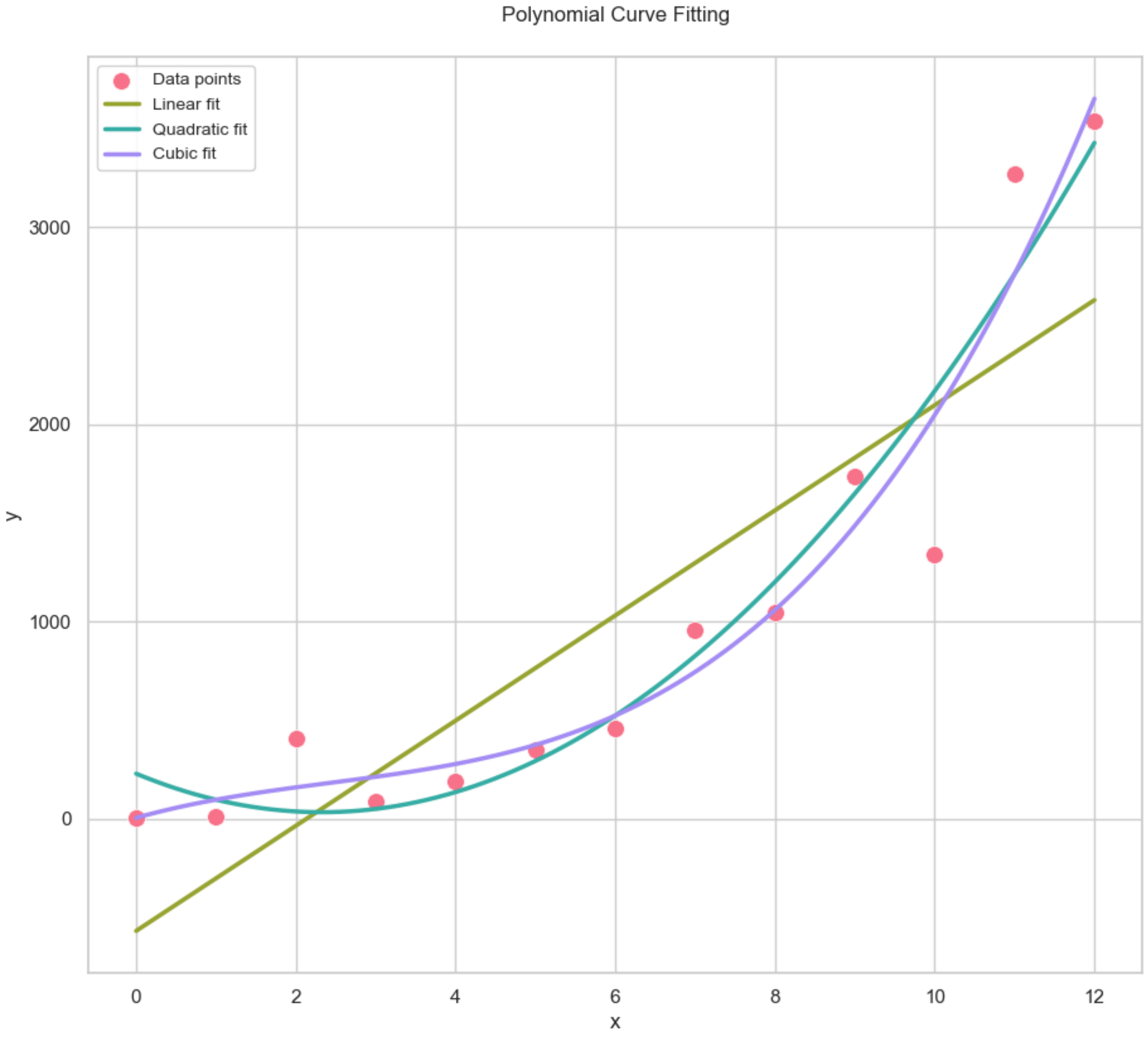
Chart Results: Two Examples

Data Source: Formula

$$y = 5 + 4x + 3x^2 + 2x^3$$



Data Source: Slightly Randomized



Text Results: Two Examples

Charts are interesting. But what is more important is the data.

Data Source: Formula

$$y = 5 + 4x + 3x^2 + 2x^3$$

Using Polynomial.fit : Linear

Coefficients : (a, b):

[-721. 306.]

Using Polynomial.fit : Quadratic

Coefficients : (a, b, c):

[137. -162. 39.]

Using Polynomial.fit : Cubic

Coefficients : (a, b, c, d):

[5. 4. 3. 2.]

Data Source: Slightly Randomized

Using Polynomial.fit : Linear

Coefficients : (a, b):

[-567.30769231 266.53846154]

Using Polynomial.fit : Quadratic

Coefficients : (a, b, c):

[229.94505495 -168.32667333 36.23876124]

Using Polynomial.fit : Cubic

Coefficients : (a, b, c, d):

[6.00274725 113.29774392 -24.83641359 3.39306527]

We can **compare** with the calculation in Excel!

That is all

Thank you for your time.