Mahidol University
Faculty of Information
and Communication Technology

Polynomial Interpolation

Rawesak Tanawongsuwan, Ph.D.
rawesak.tan@mahidol.ac.th

This slide is part of teaching materials for ITCS122 Numerical Methods
Semester 2/2023, Calendar year 2024

Interpolation

- Interpolation is the process of deriving a simple function from a set of discrete data points so that the function passes through all the given data points (i.e. reproduces the data points exactly)
- It can be used to estimate data points in-between the given ones.
- It is necessary because in science and engineering we often need to deal with discrete experimental data.
- Interpolation is also used to simplify complicated functions by sampling data points and interpolating them using a simpler function.
- Polynomials are commonly used for interpolation because they are easier to evaluate, differentiate, and integrate - known as polynomial interpolation.

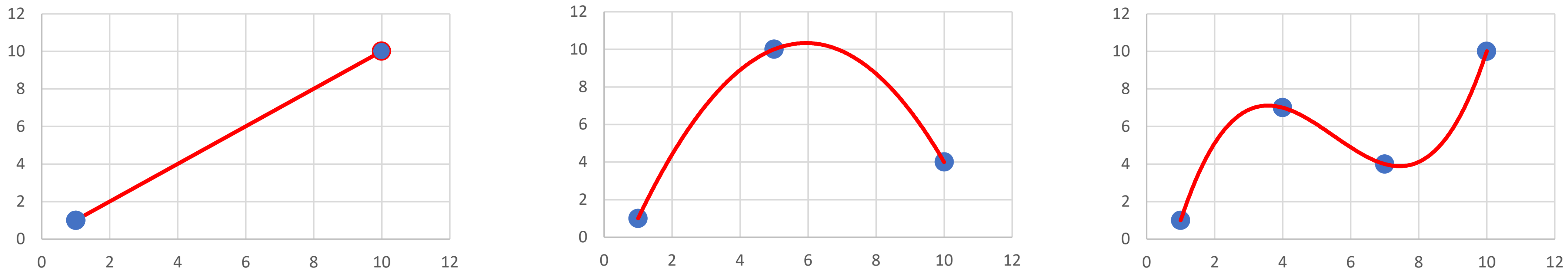
https://en.wikibooks.org/wiki/Introduction_to_Numerical_Methods/Interpolation

Polynomial interpolation

- The general formula for an $(n-1)^{th}$-order polynomial

$$f(x) = a_1 + a_2 x + a_3 x^2 + \cdots + a_n x^{n-1}$$

- For 2 points, there is only one straight line that connects them.
- For 3 points, there is only one parabola that connects them.
- For n points, there is only one $(n-1)^{th}$-order polynomial that passes through them.



Determining polynomial coefficient

- For example : determine the coefficients of the parabola, $f(x) = a_0 + a_1 x + a_2 x^2$ that passes through these three points $(3, 0.616), (4, 0.525), (5, 0.457)$

$$0.616 = a_0 + a_1(3) + a_2(3)^2$$
$$0.525 = a_0 + a_1(4) + a_2(4)^2$$
$$0.457 = a_0 + a_1(5) + a_2(5)^2$$

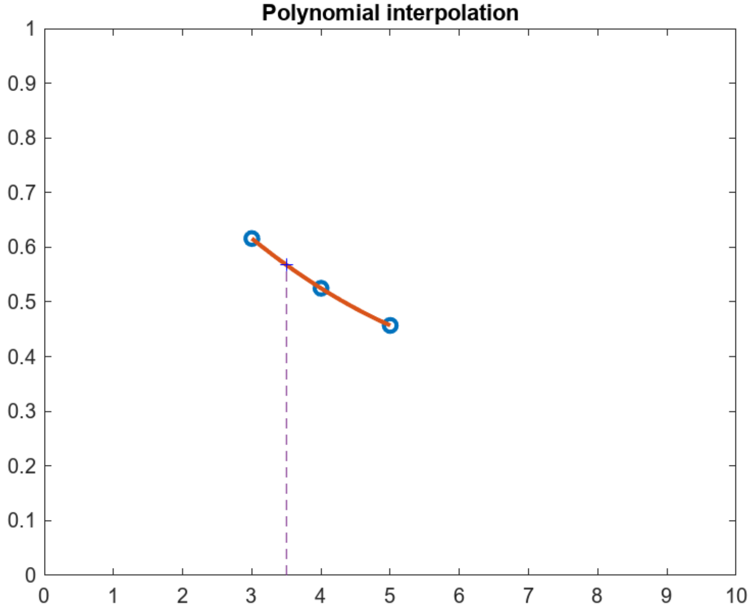
$$\begin{bmatrix} 1 & 3 & 9 \\ 1 & 4 & 16 \\ 1 & 5 & 25 \end{bmatrix} \begin{Bmatrix} a_0 \\ a_1 \\ a_2 \end{Bmatrix} = \begin{Bmatrix} 0.616 \\ 0.525 \\ 0.457 \end{Bmatrix}$$

$$a_0 = \quad 1.02700000$$
$$a_1 = -0.17150000$$
$$a_2 = \quad 0.01150000$$

$$f(x) = 1.027 - 0.1715x + 0.0115x^2$$

At $x = 3.5$

$$f(3.5) = 1.027 - 0.1715(3.5) + 0.0115(3.5)^2$$
$$= 0.567625$$



Determining polynomial coefficient

- For example : determine the coefficients of the parabola, $f(x) = a_0 + a_1 x + a_2 x^2$ that passes through these three points $(300, 0.616), (400, 0.525), (500, 0.457)$

$$0.616 = a_0 + a_1(300) + a_2(300)^2$$
$$0.525 = a_0 + a_1(400) + a_2(400)^2$$
$$0.457 = a_0 + a_1(500) + a_2(500)^2$$

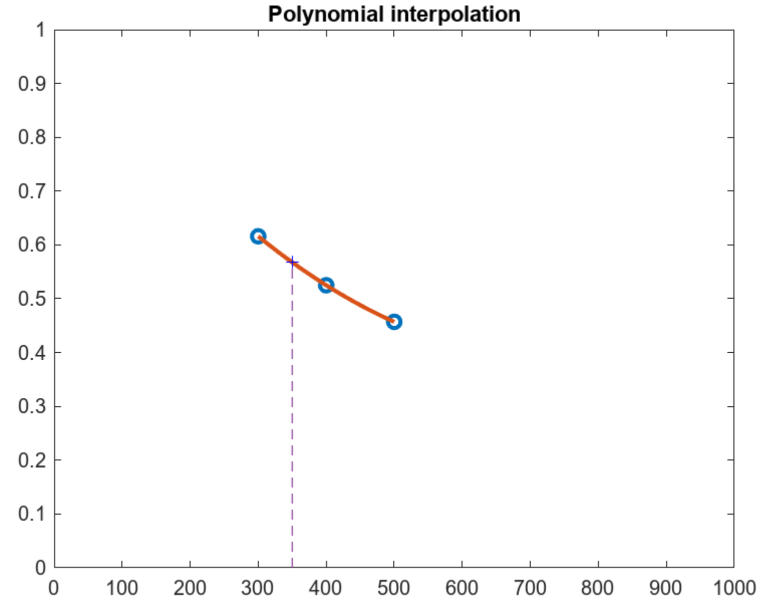
$$\begin{bmatrix} 1 & 300 & 90,00 \\ 1 & 400 & 160,000 \\ 1 & 500 & 250,000 \end{bmatrix} \begin{Bmatrix} a_0 \\ a_1 \\ a_2 \end{Bmatrix} = \begin{Bmatrix} 0.616 \\ 0.525 \\ 0.457 \end{Bmatrix}$$

$$a_0 = 1.02700000$$
$$a_1 = -0.00171500$$
$$a_2 = 0.00000115$$

$$f(x) = 1.027 - 0.001715x + 0.00000115x^2$$

At $x = 350$

$$f(350) = 1.027 - 0.001715(350) + 0.00000115(350)^2$$
$$= 0.567625$$



Determining polynomial coefficient

- For example : determine the coefficients of the parabola, $f(x) = a_0 + a_1 x + a_2 x^2$ that passes through these three points $(300, 0.616), (400, 0.525), (500, 0.457)$

$$0.616 = a_0 + a_1(300) + a_2(300)^2$$
$$0.525 = a_0 + a_1(400) + a_2(400)^2$$
$$0.457 = a_0 + a_1(500) + a_2(500)^2$$

$$\begin{bmatrix} 1 & 300 & 90,00 \\ 1 & 400 & 160,000 \\ 1 & 500 & 250,000 \end{bmatrix} \begin{Bmatrix} a_0 \\ a_1 \\ a_2 \end{Bmatrix} = \begin{Bmatrix} 0.616 \\ 0.525 \\ 0.457 \end{Bmatrix}$$

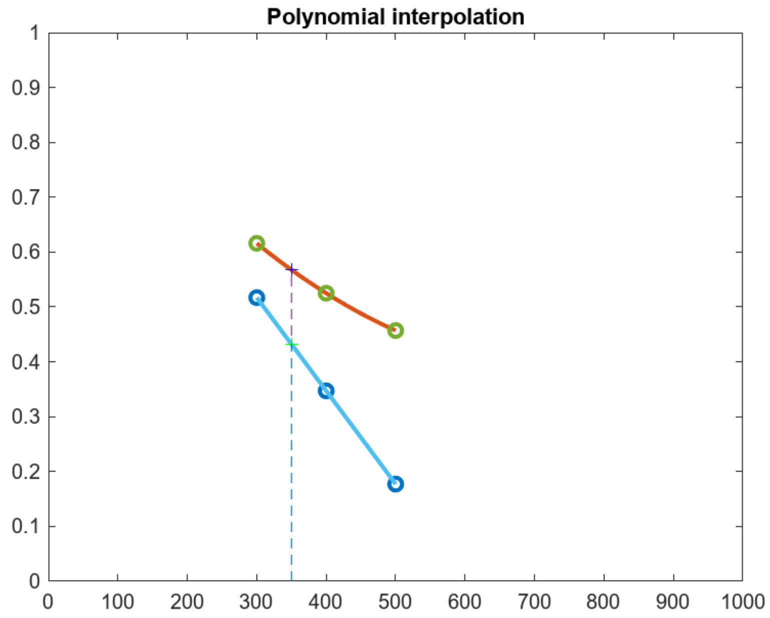
$$a_0 = 1.02700000 \approx 1.0270$$
$$a_1 = -0.00171500 \approx -0.0017$$
$$a_2 = 0.00000115 \approx 0.0000$$

$$f(x) = 1.027 - 0.0017x + 0.0000x^2$$

Note that this is linear!!!

At $x = 350$

$$f(350) = 1.027 - 0.0017\,(350) + 0.0000\,(350)^2$$
$$= 0.4320$$

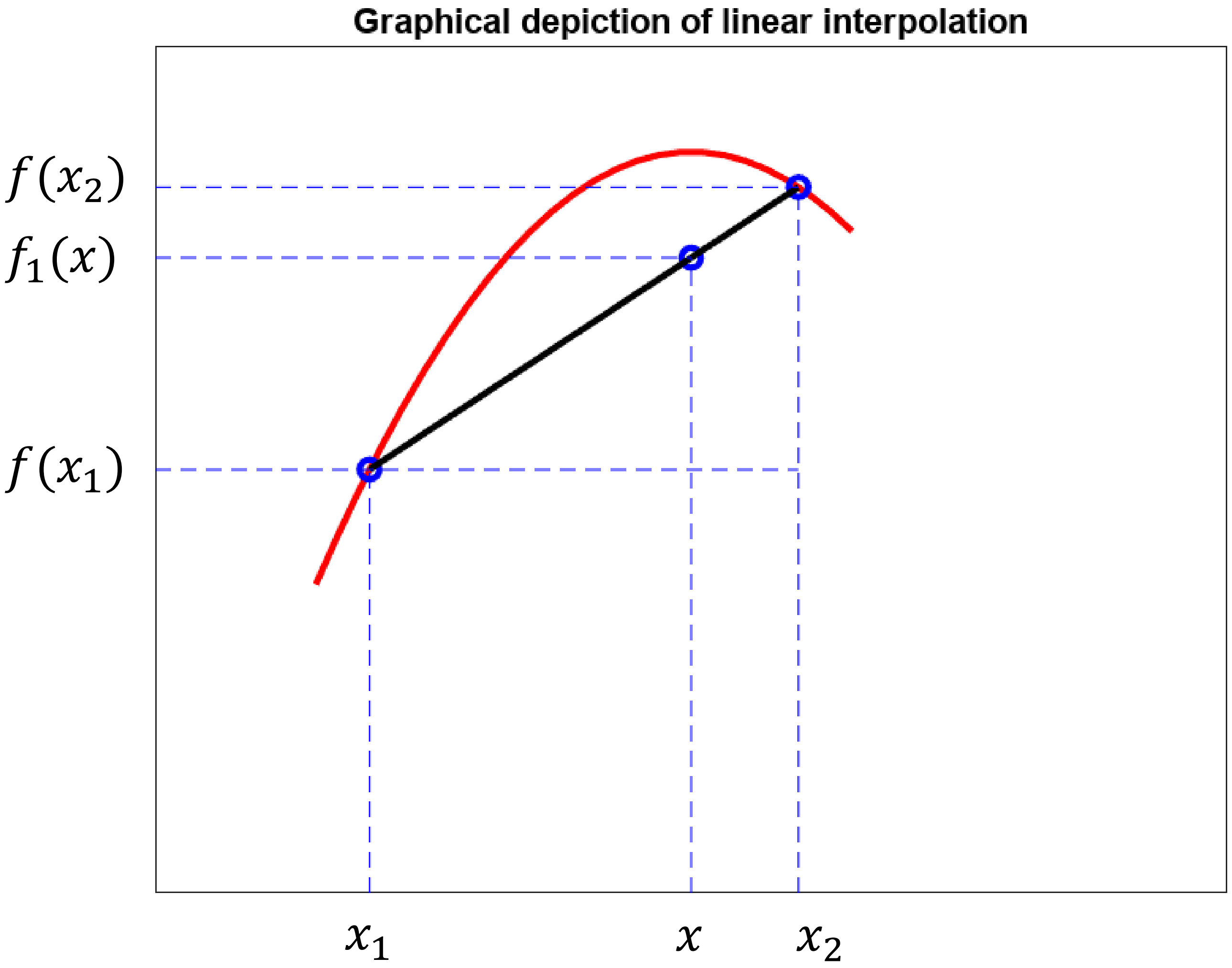


Flaw in determining polynomial coefficients with simultaneous equations

$$\begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ 1 & x_3 & x_3^2 \end{bmatrix} \begin{Bmatrix} a_0 \\ a_1 \\ a_2 \end{Bmatrix} = \begin{Bmatrix} f(x_1) \\ f(x_1) \\ f(x_1) \end{Bmatrix}$$

- Coefficient matrices of this form are referred to as *Vandermonde* matrices.
- Such matrices are very ill-conditioned which means their solutions are very sensitive to round off errors.
- There are alternative approaches that do not appear to have this flaw such as the Newton and the Lagrange polynomials.

Newton's interpolating polynomial
→ Linear interpolation (simplest form)



Newton’s interpolating polynomial → Linear interpolation (simplest form)

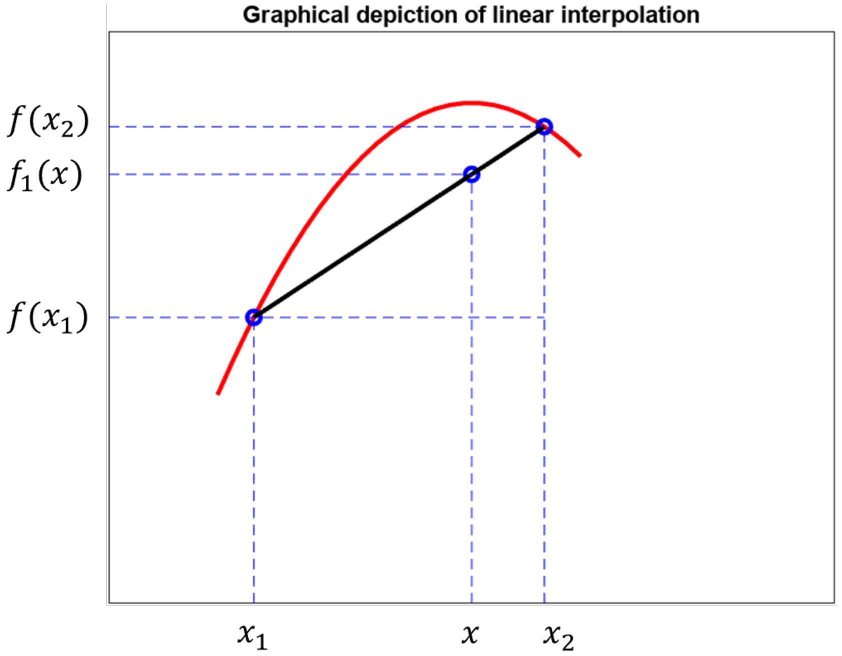
Connect two data points with a straight line : *linear interpolation*

$$\frac{f_1(x) - f(x_1)}{x - x_1} = \frac{f(x_2) - f(x_1)}{x_2 - x_1}$$

$$f_1(x) = f(x_1) + \frac{f(x_2) - f(x_1)}{x_2 - x_1}(x - x_1)$$

$f_1(x)$: a first-order interpolating polynomial

$\frac{f_1(x_2)-f(x_1)}{x_2-x_1}$: a finite-difference approximation of the first derivative



Example : Linear interpolation

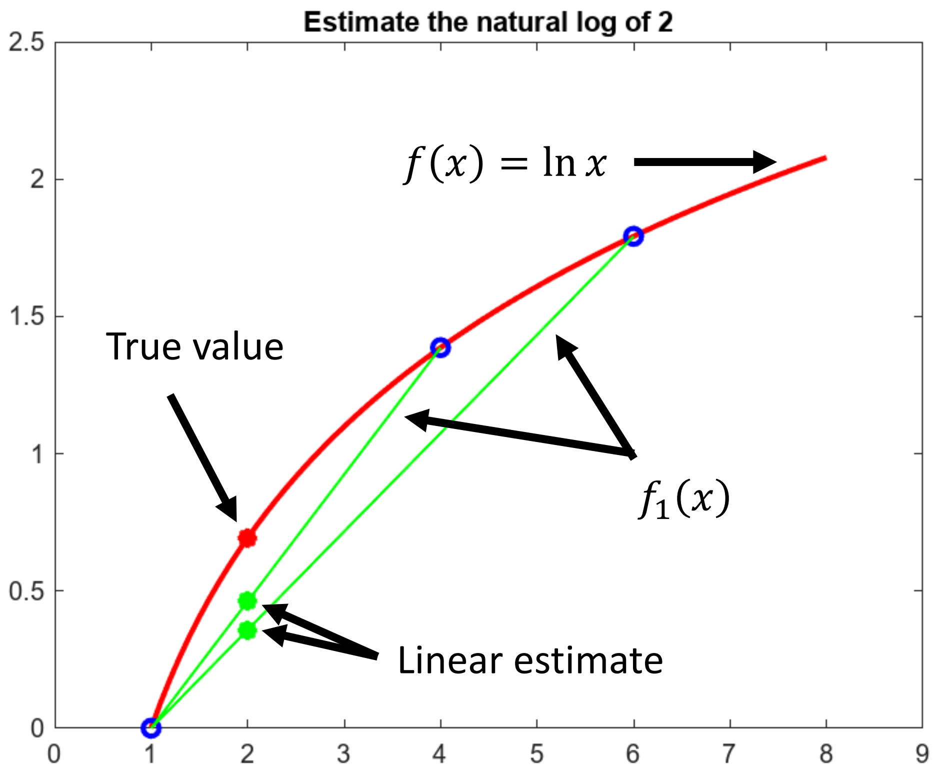
Estimate the natural logarithm of 2 using linear interpolation.

1) Interpolating between $\ln 1$ and $\ln 6$

2) Interpolating between $\ln 1$ and $\ln 4$

$$\ln 1 = 0$$
$$\ln 4 = 1.386294$$
$$\ln 6 = 1.791759$$

True value of $\ln 2 = 0.6931472$



Example : Linear interpolation

Estimate the natural logarithm of 2 using linear interpolation.

1) Interpolating between $\ln 1$ and $\ln 6$

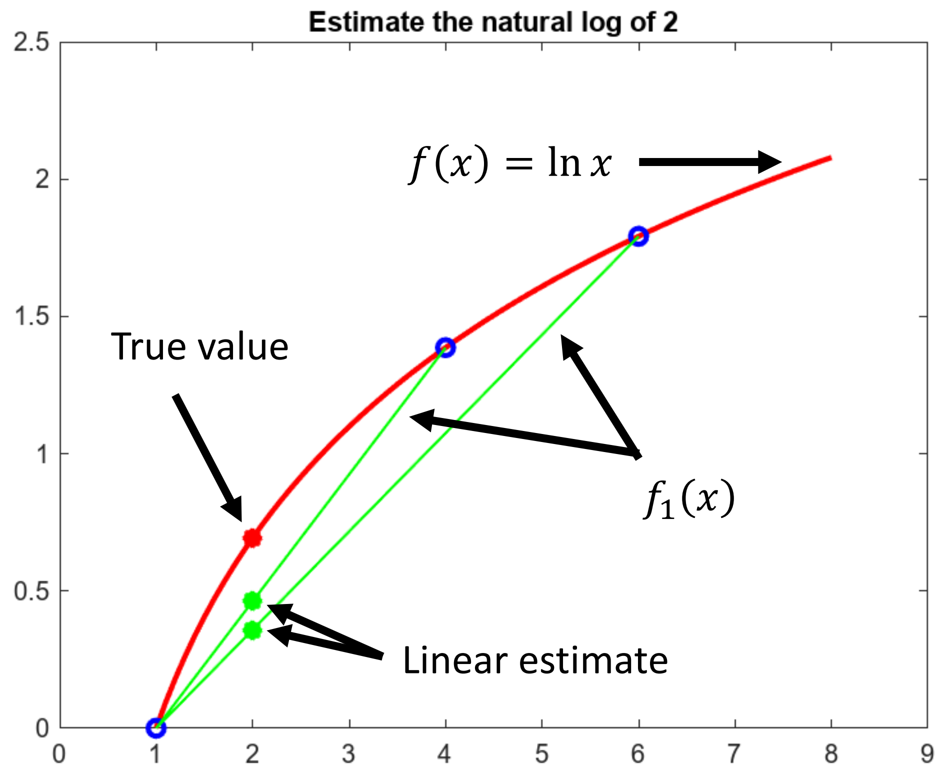
$f_1(2) =$

$\varepsilon_t =$

2) Interpolating between $\ln 1$ and $\ln 4$

$f_1(2) =$

$\varepsilon_t =$



The smaller the interval between the data points, the better the approximation

Newton’s interpolating polynomial → Quadratic interpolation

- We can improve the estimate by introducing some curvature into the line connecting the points.

- If 3 data points are available, we can use second-order polynomial (a quadratic polynomial or a parabola).

$$f_2(x) = b_1 + b_2(x - x_1) + b_3(x - x_1)(x - x_2)$$

Newton’s interpolating polynomial → Quadratic interpolation

- When $x = x_1$

$$b_1 = f(x_1)$$

- When $x = x_2$

$$b_2 = \frac{f(x_2) - f(x_1)}{x_2 - x_1}$$

- When $x = x_3$

$$b_3 = \frac{\dfrac{f(x_3) - f(x_2)}{x_3 - x_2} - \dfrac{f(x_2) - f(x_1)}{x_2 - x_1}}{x_3 - x_1}$$

b_2 : a finite-difference approximation of the first derivative

b_3 : a finite-difference approximation of the second derivative

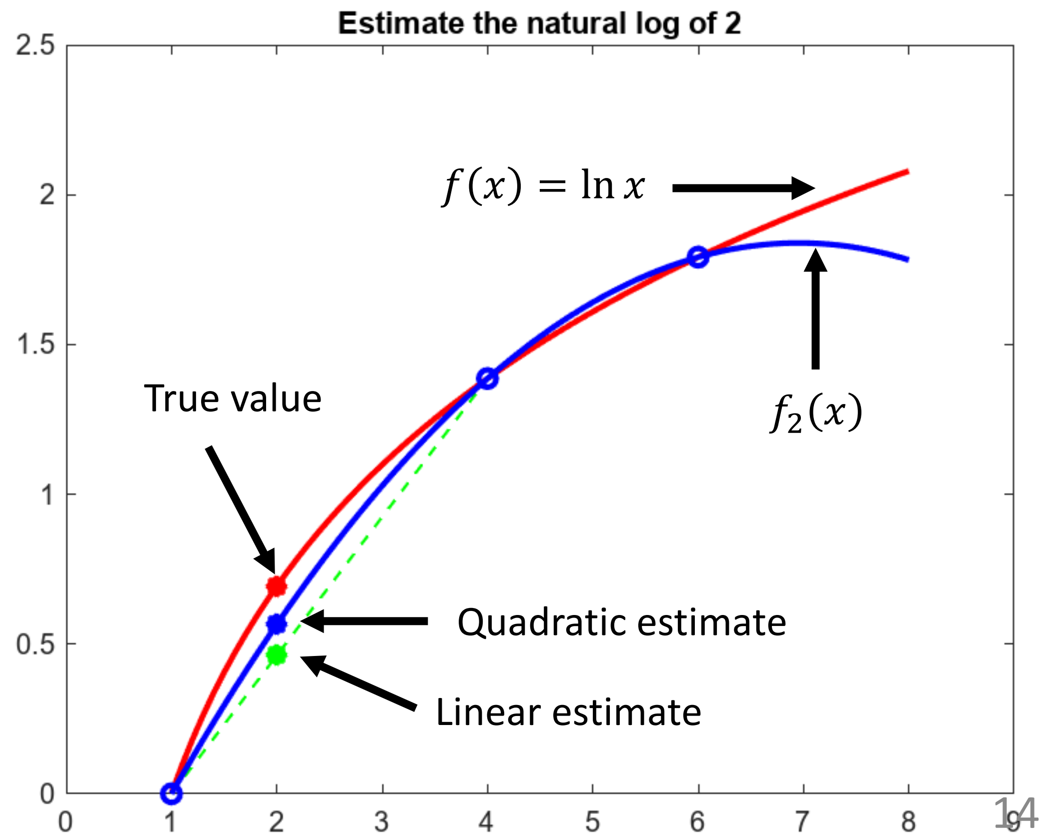
Example : Quadratic interpolation

Estimate the natural logarithm of 2 using quadratic interpolation.

1) Interpolating between $\ln 1$, $\ln 4$ and $\ln 6$

$$\ln 1 = 0$$
$$\ln 4 = 1.386294$$
$$\ln 6 = 1.791759$$

True value of $\ln 2 = 0.6931472$



Example : Quadratic interpolation

Interpolating between $\ln 1$, $\ln 4$ and $\ln 6$

$x_1 = 1 \quad f(x_1) = 0$
$x_2 = 4 \quad f(x_2) = 1.386294$
$x_3 = 6 \quad f(x_3) = 1.791759$

$b_1 =$

$b_2 =$

$b_3 =$

Example : Quadratic interpolation

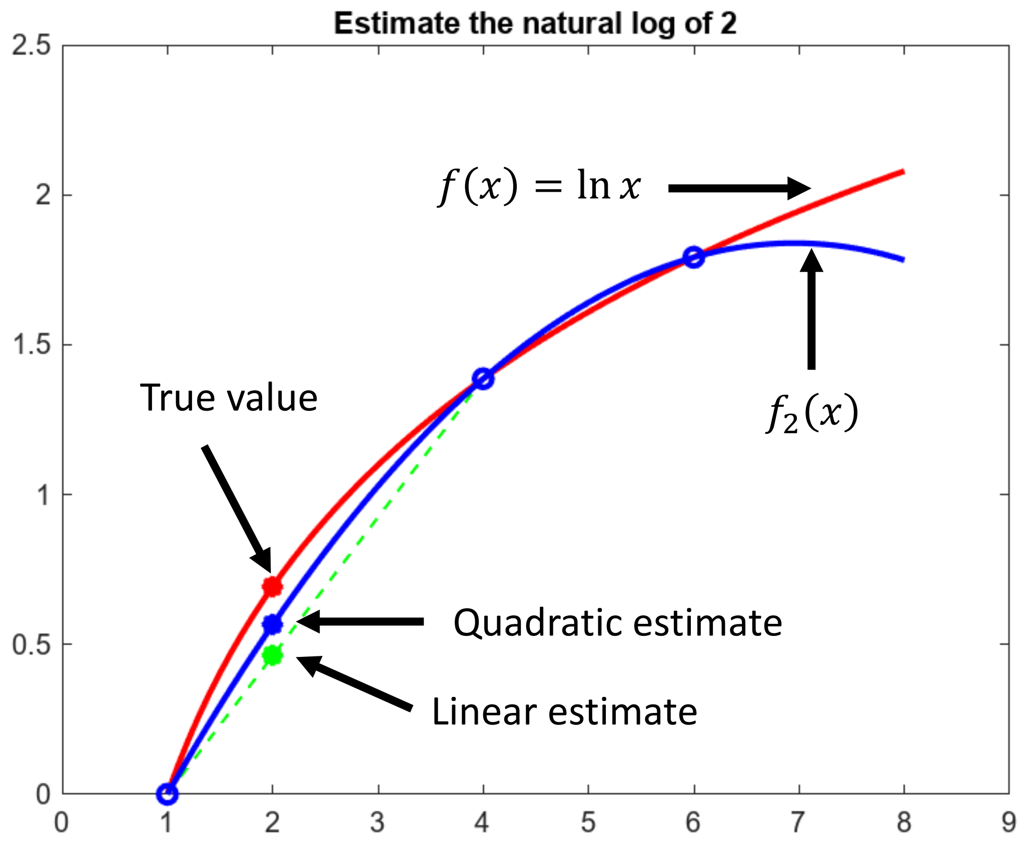
Quadratic formula $\boldsymbol{f_2(x) =}$

Evaluate at $x = 2$

$f_2(2) =$

Relative error

$\varepsilon_t =$



General form of Newton’s interpolating polynomials

- The $(n-1)^{th}$-order polynomial

$$f_{n-1}(x) = b_1 + b_2(x - x_1) + \cdots + b_n(x - x_1)(x - x_2) \cdots (x - x_{n-1})$$

- For an $(n-1)^{th}$-order polynomial, n data points are required :

$$[x_1, f(x_1)], [x_2, f(x_2)], \dots [x_n, f(x_n)]$$

- The coefficients can be evaluated by

$$\begin{gathered} b_1 = f(x_1) \\ b_2 = f[x_2, x_1] \\ b_2 = f[x_3, x_2, x_1] \\ \vdots \\ b_n = f[x_n, x_{n-1}, \dots, x_2, x_1] \end{gathered}$$

Where the bracketed function evaluations are finite divided difference.

Finite divided difference

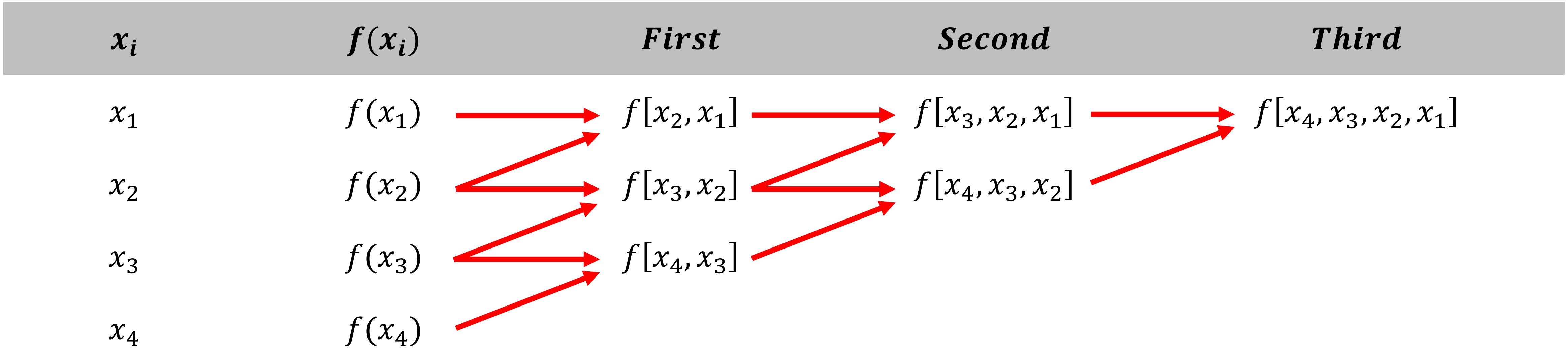
$$f\left[x_i, x_j\right] = \frac{f(x_i) - f\left(x_j\right)}{x_i - x_j}$$

$$f\left[x_i, x_j, x_k\right] = \frac{f[x_i, x_j] - f[x_j, x_k]}{x_i - x_k}$$

$$\vdots$$

$$f[x_n, x_{n-1}, \dots, x_2, x_1] = \frac{f[x_n, x_{n-1}, \dots, x_2] - f[x_{n-1}, x_{n-2}, \dots, x_1]}{x_n - x_1}$$

Recursive nature of finite divided difference



$\boldsymbol{x_i}$	$\boldsymbol{f(x_i)}$	***First***	***Second***	***Third***
x_1	$f(x_1)$	$f[x_2, x_1]$	$f[x_3, x_2, x_1]$	$f[x_4, x_3, x_2, x_1]$
x_2	$f(x_2)$	$f[x_3, x_2]$	$f[x_4, x_3, x_2]$	
x_3	$f(x_3)$	$f[x_4, x_3]$		
x_4	$f(x_4)$			

General form of Newton’s interpolating polynomial

$$
\begin{aligned}
f_{n-1}(x) = f(x_1) &+ (x - x_1) f[x_2, x_1] + (x - x_1)(x - x_2) f[x_3, x_2, x_1] \\
&+ \cdots + (x - x_1)(x - x_2) \cdots (x - x_{n-1})\, f[x_n, x_{n-1}, \ldots, x_2, x_1]
\end{aligned}
$$

Observations

1) It is not necessary that the data points be equally spaced or that the abscissa values (x values) be in ascending order or sorted order. However, they should be ordered so that they are centered around and as close as possible to the unknown.

2) Coefficients can be solved recursively, that is higher-order differences are computed by taking differences of lower-order differences. We can implement the method using recursive programming.

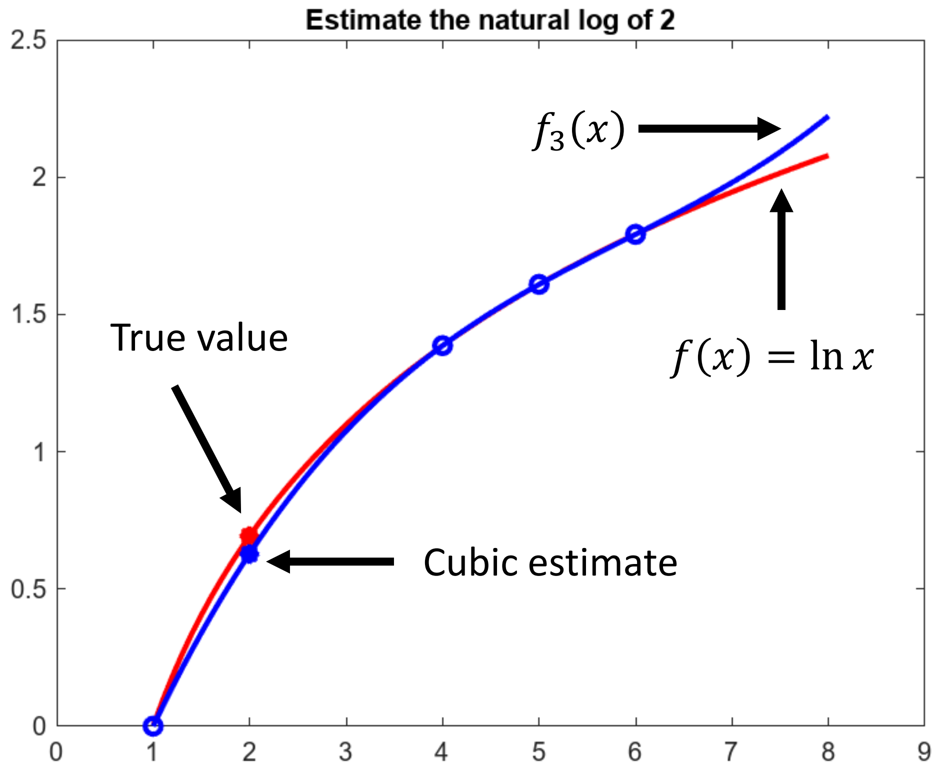
Example : Cubic interpolation

Estimate the natural logarithm of 2 using quadratic interpolation.

1) Interpolating between $\ln 1$, $\ln 4$, $\ln 6$, and $\ln 5$

$\ln 1 = 0$
$\ln 4 = 1.386294$
$\ln 6 = 1.791759$
$\ln 5 = 1.609438$

True value of $\ln 2 = 0.6931472$



Example : Cubic interpolation

Interpolating between $\ln 1$, $\ln 4$, $\ln 6$ and $\ln 5$

$x_1 = 1 \quad f(x_1) = 0$
$x_2 = 4 \quad f(x_2) = 1.386294$
$x_3 = 6 \quad f(x_3) = 1.791759$
$x_3 = 5 \quad f(x_4) = 1.609438$

The third-order polynomial with $n = 4$

$$f_3(x) = b_1 + b_2(x - x_1) + b_3(x - x_1)(x - x_2) + b_4(x - x_1)(x - x_2)(x - x_3)$$

Example : Cubic interpolation

The first divided differences

$$b_2 = f[x_2, x_1] = \frac{1.386294 - 0}{4 - 1} = 0.4620981$$

$$f[x_3, x_2] = \frac{1.791759 - 1.386294}{6 - 4} = 0.2027326$$

$$f[x_4, x_3] = \frac{1.609438 - 1.791759}{5 - 6} = 0.1823216$$

The second divided differences

$$b_3 = f[x_3, x_2, x_1] = \frac{0.2027326 - 0.4620981}{6 - 1} = -0.05187311$$

$$f[x_4, x_3, x_2] = \frac{0.1823216 \ - 0.2027326}{5 - 4} = -0.02041100$$

The third divided difference

$$b_4 = f[x_4, x_3, x_2, x_1] = \frac{-0.02041100 - (-0.05187311)}{5 - 1} = 0.007865529$$

Example : Cubic interpolation

x_i	$f(x_i)$	***First***	***Second***	***Third***
1	0			
4	1.386294			
6	1.791759			
5	1.609438			

Example : Cubic interpolation

The interpolating cubic

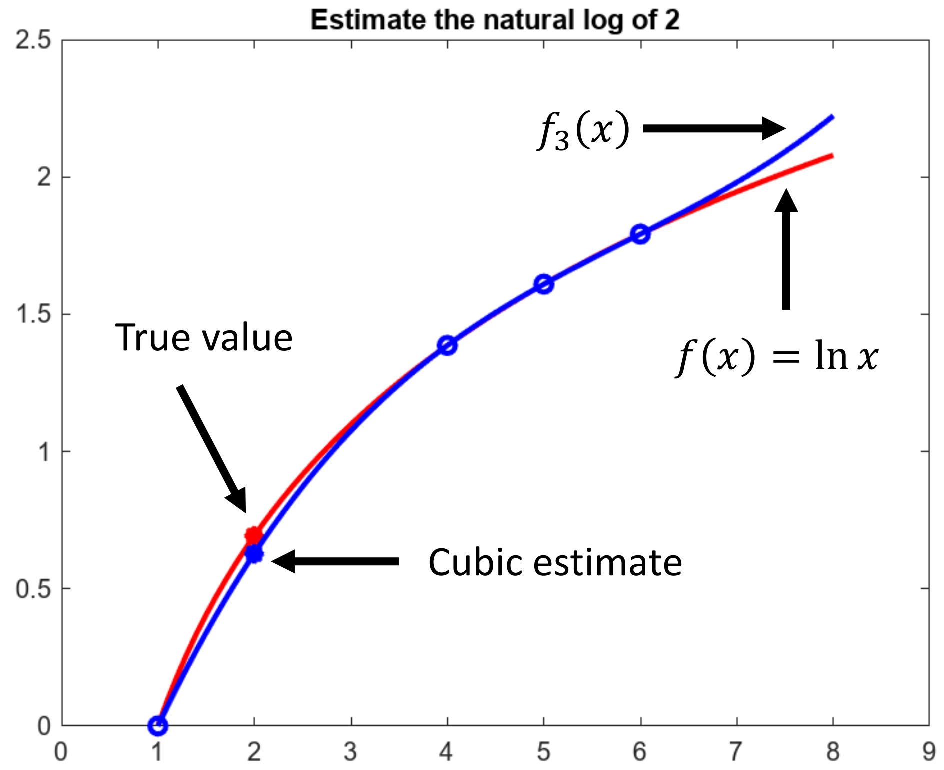
$f_3(x) =$

Evaluate at $x = 2$

$f_3(2) =$

Relative error

$\varepsilon_t =$



Estimate the natural log of 2

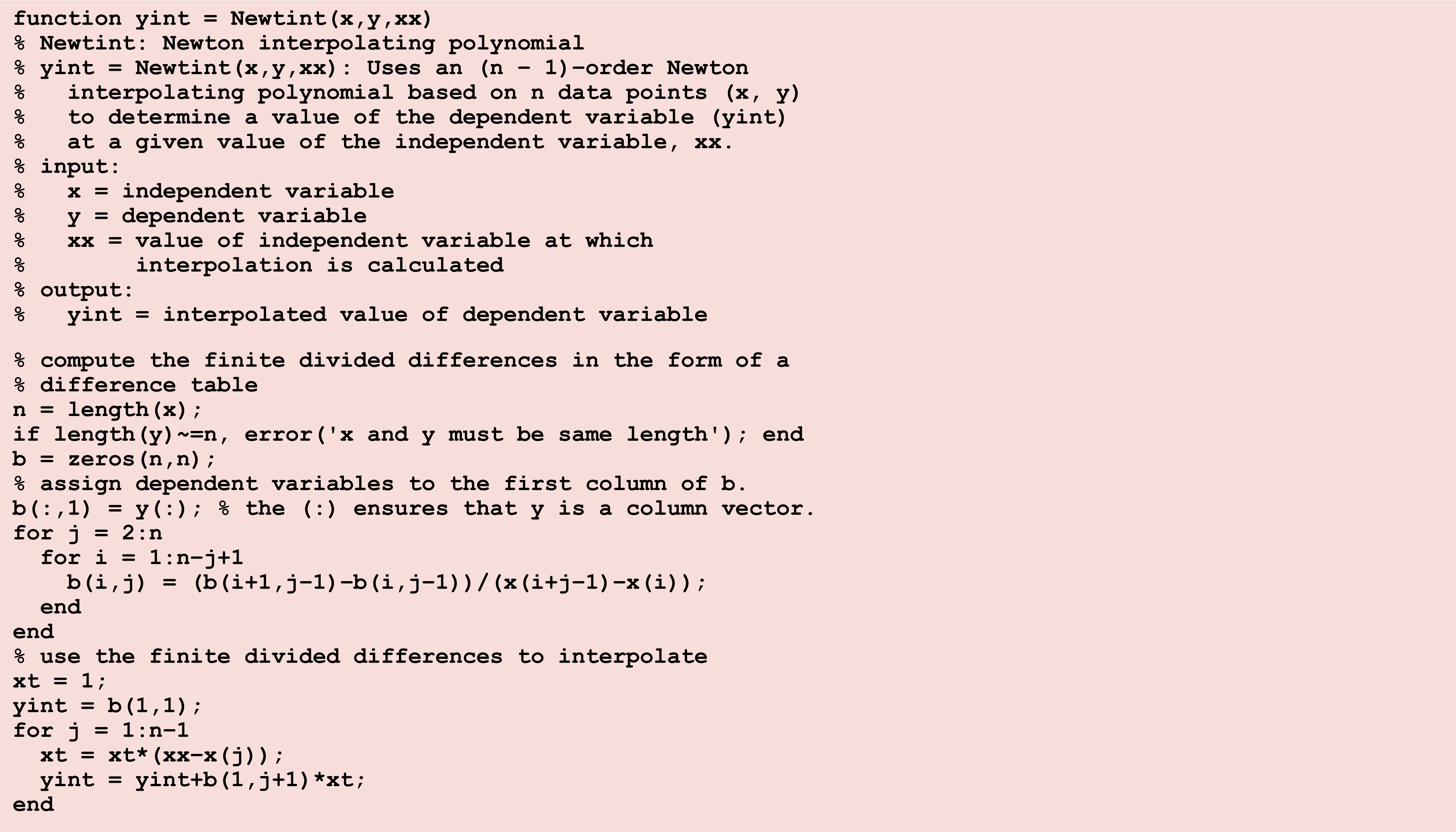
$f_3(x)$

$f(x) = \ln x$

True value

Cubic estimate

MATLAB Implementation 1



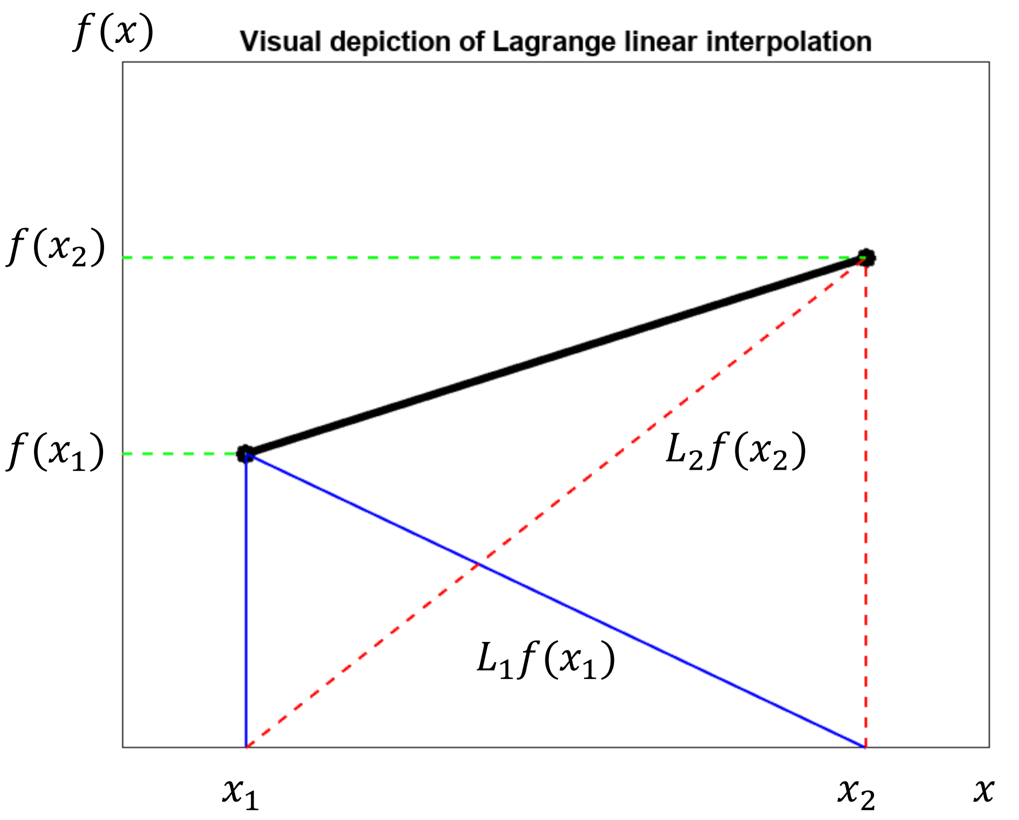
```
function yint = Newtint(x,y,xx)
% Newtint: Newton interpolating polynomial
% yint = Newtint(x,y,xx): Uses an (n - 1)-order Newton
%   interpolating polynomial based on n data points (x, y)
%   to determine a value of the dependent variable (yint)
%   at a given value of the independent variable, xx.
% input:
%   x = independent variable
%   y = dependent variable
%   xx = value of independent variable at which
%        interpolation is calculated
% output:
%   yint = interpolated value of dependent variable

% compute the finite divided differences in the form of a
% difference table
n = length(x);
if length(y)~=n, error('x and y must be same length'); end
b = zeros(n,n);
% assign dependent variables to the first column of b.
b(:,1) = y(:); % the (:) ensures that y is a column vector.
for j = 2:n
  for i = 1:n-j+1
    b(i,j) = (b(i+1,j-1)-b(i,j-1))/(x(i+j-1)-x(i));
  end
end
% use the finite divided differences to interpolate
xt = 1;
yint = b(1,1);
for j = 1:n-1
  xt = xt*(xx-x(j));
  yint = yint+b(1,j+1)*xt;
end
```

Linear Lagrange interpolating polynomial

Formulate a linear interpolating polynomial as the weighted average of the two values that we are connecting by a straight line

$$f_1(x) = L_1 f(x_1) + L_2 f(x_2)$$



The L's are the weighting coefficients

$L_1 = \frac{x-x_2}{x_1-x_2}$ 🡐 The first coefficient L_1 is the straight line

🡐 when $x = x_1$, $L_1 = 1$

🡐 when $x = x_2$, $L_1 = 0$

$L_2 = \frac{x-x_1}{x_2-x_1}$ 🡐 The second coefficient L_2 is the straight line

🡐 when $x = x_2$, $L_2 = 1$

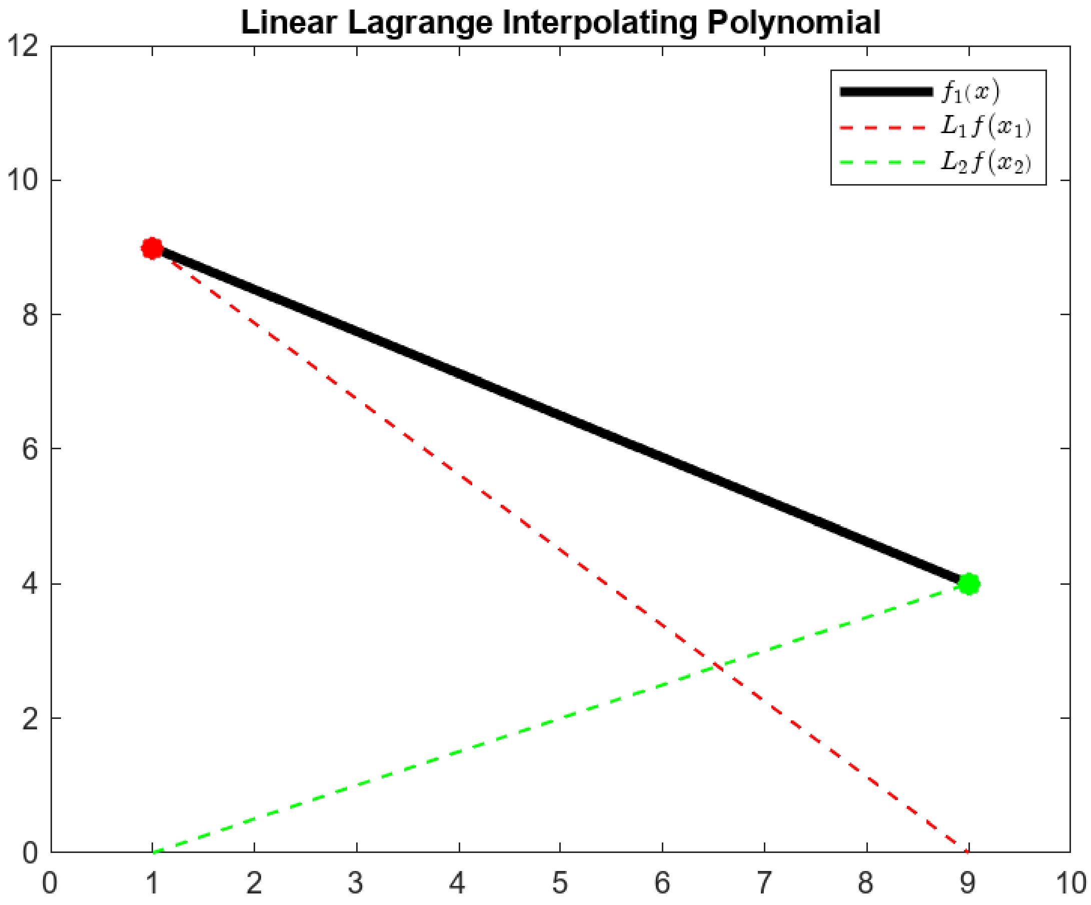
🡐 when $x = x_1$, $L_2 = 0$

$f_1(x) = \frac{x-x_2}{x_1-x_2} f(x_1) + \frac{x-x_1}{x_2-x_1} f(x_2)$ 🡐 $f_1(x)$ is the straight line that connects the points

Linear Lagrange interpolating polynomial

- This image shows, for two points $((1,9),(9,4))$, the (linear) interpolation polynomial $f_1(x)$ (solid, black), which is the sum of the scaled basis polynomials $L_1 f(x_1)$ and $L_2 f(x_2)$.

- The interpolation polynomial passes through all two control points, and each scaled basis polynomial passes through its respective control point and is 0 where x corresponds to the other control points.



Quadratic Lagrange interpolating polynomial

Fitting a parabola through 3 points.

3 parabolas would be used with each one passing through one of the points and equaling zero at the other two.

Their sum would represent the unique parabola that connects the three points

Formulate a quadratic Lagrange interpolating polynomial as

$$f_2(x) = L_1 f(x_1) + L_2 f(x_2) + L_3 f(x_3)$$

The L's are the weighting coefficients

$L_1 = \frac{(x-x_2)(x-x_3)}{(x_1-x_2)(x_1-x_3)}$ 🡐 The first coefficient L_1 is the parabola

🡐 when $x = x_1, \ \ L_1 = 1$ when $x = x_2, \ \ L_1 = 0$ when $x = x_3, \ \ L_1 = 0$

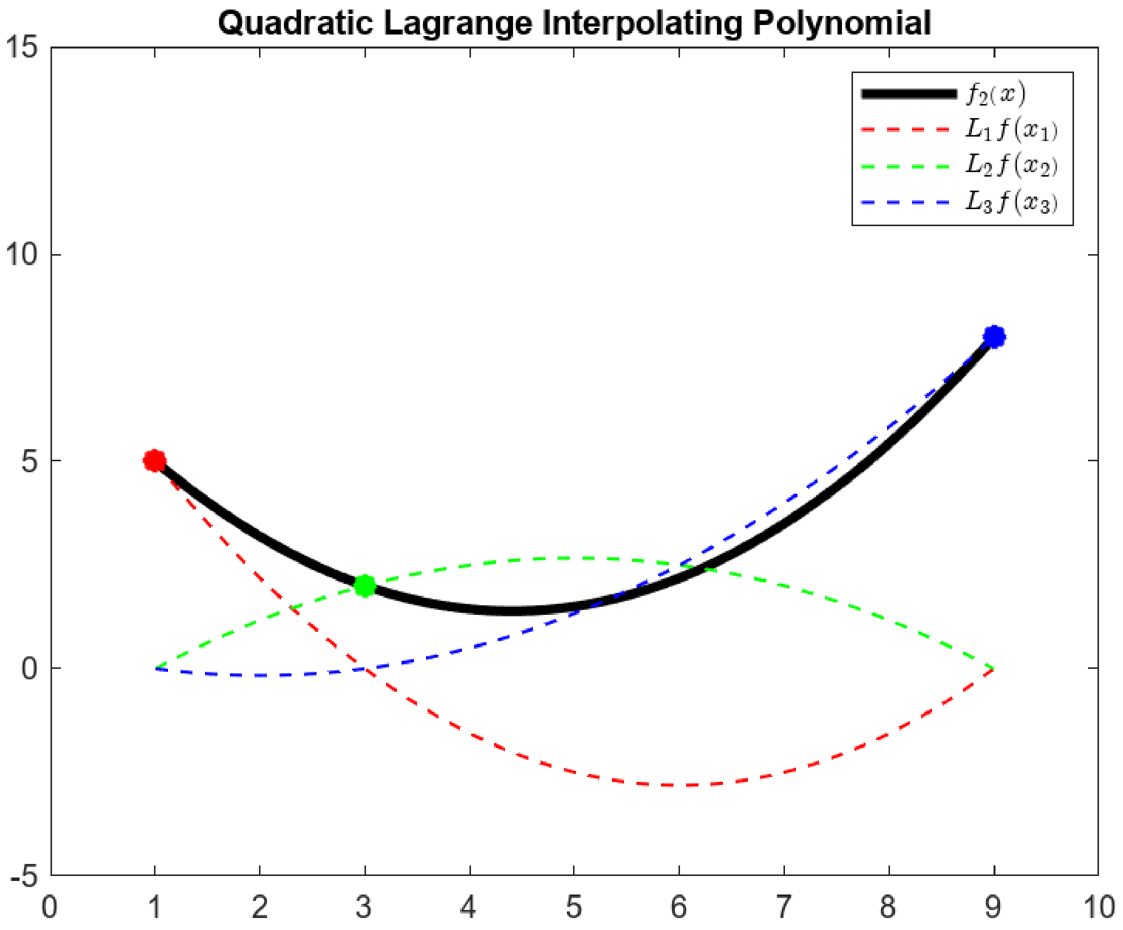
L_2 and L_3 work in a similar fashion

$$f_2(x) = \frac{(x-x_2)(x-x_3)}{(x_1-x_2)(x_1-x_3)} f(x_1) + \frac{(x-x_1)(x-x_3)}{(x_2-x_1)(x_2-x_3)} f(x_2) + \frac{(x-x_1)(x-x_2)}{(x_3-x_1)(x_3-x_2)} f(x_3)$$

$f_2(x)$ is the parabola that connects three points

Quadratic Lagrange interpolating polynomial

- This image shows, for three points $((1,5),(3,3),(9,8))$, the (quadratic) interpolation polynomial $f_2(x)$ (solid, black), which is the sum of the scaled basis polynomials $L_1 f(x_1)$, $L_2 f(x_2)$, $L_3 f(x_3)$.
- The interpolation polynomial passes through all three control points, and each scaled basis polynomial passes through its respective control point and is 0 where x corresponds to the other control points.



Cubic Lagrange interpolating polynomial

Fitting a cubic polynomial through 4 points.

4 cubic polynomials would be used with each one passing through one of the points and equaling zero at the other two.

Their sum would represent the unique cubic polynomial that connects the four points

Formulate a cubic Lagrange interpolating polynomial as

$$f_3(x) = L_1 f(x_1) + L_2 f(x_2) + L_3 f(x_3) + L_4 f(x_4)$$

The L's are the weighting coefficients

$$f_3(x) = \frac{(x-x_2)(x-x_3)(x-x_4)}{(x_1-x_2)(x_1-x_3)(x_1-x_4)} f(x_1) + \frac{(x-x_1)(x-x_3)(x-x_4)}{(x_2-x_1)(x_2-x_3)(x_2-x_4)} f(x_2)$$

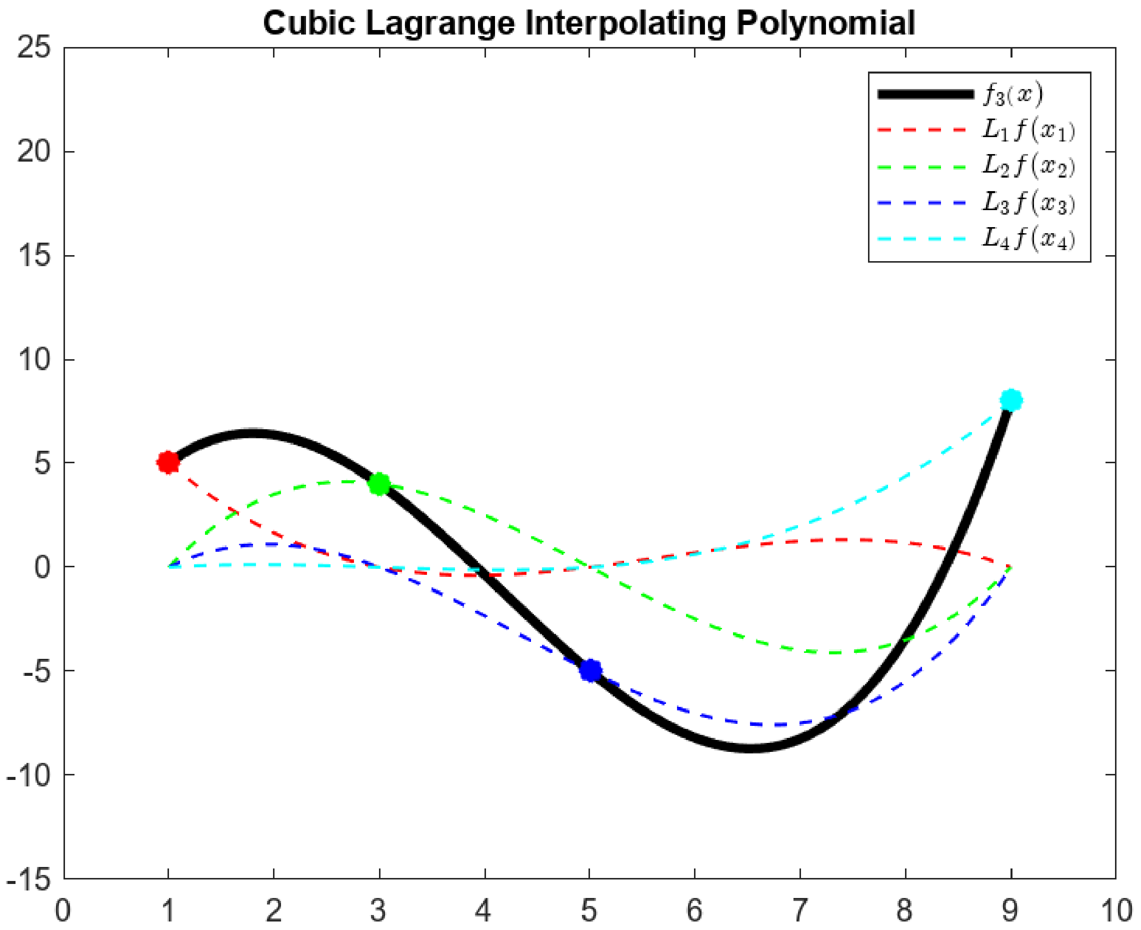
$$+ \frac{(x-x_1)(x-x_2)(x-x_4)}{(x_3-x_1)(x_3-x_2)(x_3-x_4)} f(x_3) + \frac{(x-x_1)(x-x_2)(x-x_3)}{(x_4-x_1)(x_4-x_2)(x_4-x_3)} f(x_4)$$

$f_3(x)$ is the cubic polynomial that connects four points

Cubic Lagrange interpolating polynomial

- This image shows, for four points $((1,5),(3,4),(5,-5),(9,8))$, the (quadratic) interpolation polynomial $f_2(x)$ (solid, black), which is the sum of the scaled basis polynomials $L_1 f(x_1)$, $L_2 f(x_2)$, $L_3 f(x_3)$, and $L_4 f(x_4)$.

- The interpolation polynomial passes through all four control points, and each scaled basis polynomial passes through its respective control point and is 0 where x corresponds to the other control points.



Higher-order Lagrange interpolating polynomial

Fitting $(n-1)$th-order polynomial through n points.

n cubic polynomials would be used with each one passing through one of the points and equaling zero at the other two.

Their sum would represent the unique $(n-1)$th-order polynomial that connects the n points

Formulate a $(n-1)$th-order Lagrange interpolating polynomial as

$$f_{n-1}(x) = \sum_{i=1}^{n} L_i(x) f(x_i)$$

where L's are the weighting coefficients, n = the number of data points, and Π represents the "product of"

$$L_i(x) = \prod_{\substack{j=1 \\ j \neq i}}^{n} \frac{x - x_j}{x_i - x_j}$$

Example : Lagrange interpolating polynomial

- Use a Lagrange interpolating polynomial of the first and second order to evaluate $f(15)$ based on the following data :

$x_1 = 0$	$x_2 = 20$	$x_3 = 40$
$f(x_1) = 3.85$	$f(x_2) = 0.800$	$f(x_3) = 0.212$

The first-order polynomial

$f_1(x) =$

The second-order polynomial

$f_2(x)$

MATLAB Implementation 2

```
function yint = Lagrange(x,y,xx)
% Lagrange: Lagrange interpolating polynomial
%   yint = Lagrange(x,y,xx): Uses an (n - 1)-order
%     Lagrange interpolating polynomial based on n data points
%     to determine a value of the dependent variable (yint) at
%     a given value of the independent variable, xx.
% input:
%   x = independent variable
%   y = dependent variable
%   xx = value of independent variable at which the
%        interpolation is calculated
% output:
%   yint = interpolated value of dependent variable

n = length(x);
if length(y)~=n, error('x and y must be same length'); end
s = 0;
for i = 1:n
  product = y(i);
  for j = 1:n
   if i ~= j
     product = product*(xx-x(j))/(x(i)-x(j));
   end
  end
  s = s+product;
end
yint = s;
```

Inverse interpolation

- In the context of interpolaton, we are given x and $f(x)$ values. The values of the x's are typically uniformly spaced.

$\boldsymbol{x}$	1	2	3	4	5	6	7
$\boldsymbol{f(x)}$	1	0.5	0.3333	0.25	0.2	0.1667	0.1429

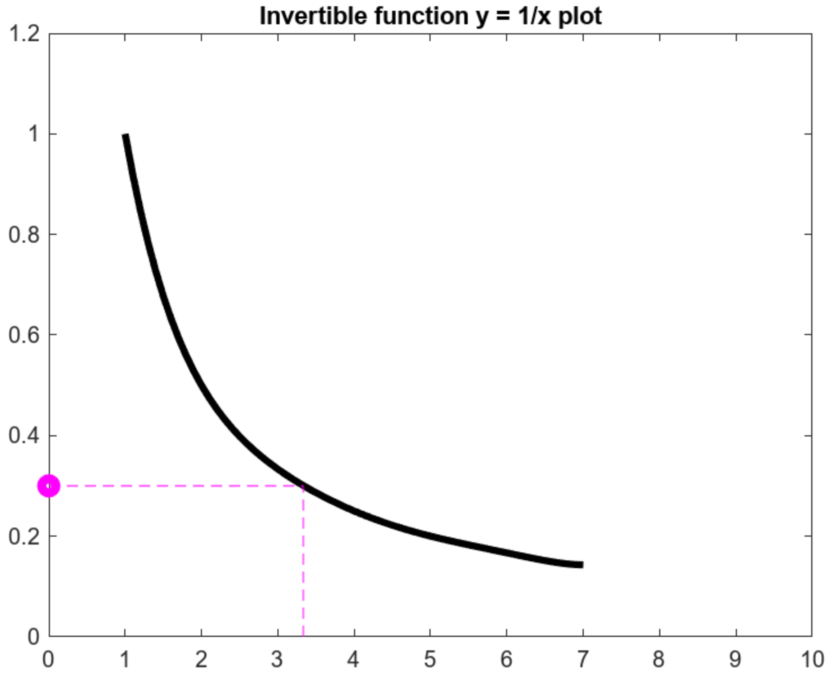
- We then usually want to know, for example, when value of $x = 3.5$, what value of $f(x) = ?$

- However, what if we are given a value for $f(x) = 0.3$ and we must find the corresponding value of $x = ?$.

Case 1 : Invertible function

- Let’s deal with the case that the underlying model is an invertible function.

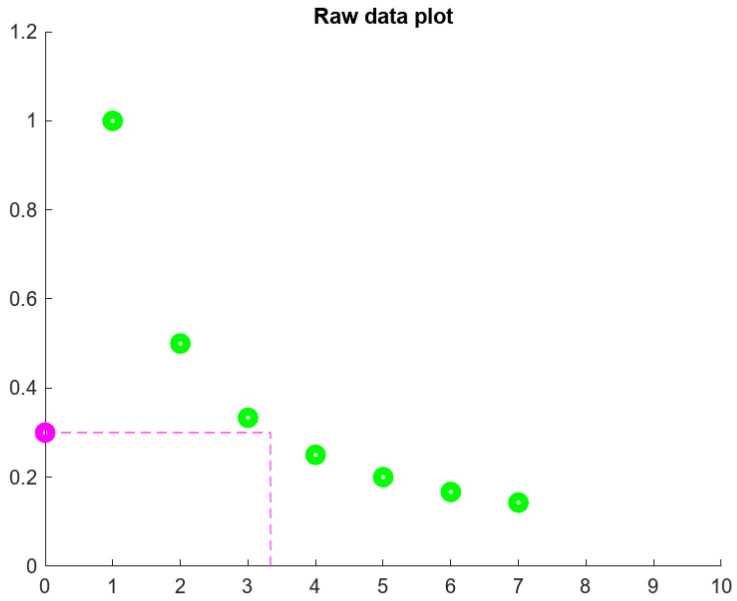
- For example, for the invertible function $f(x) = \frac{1}{x}$, if we want to know what value of x when $f(x) = 0.3$, then we can easily solve $0.3 = \frac{1}{x} \rightarrow x = \frac{1}{0.3} = 3.3333$



Inverse interpolation

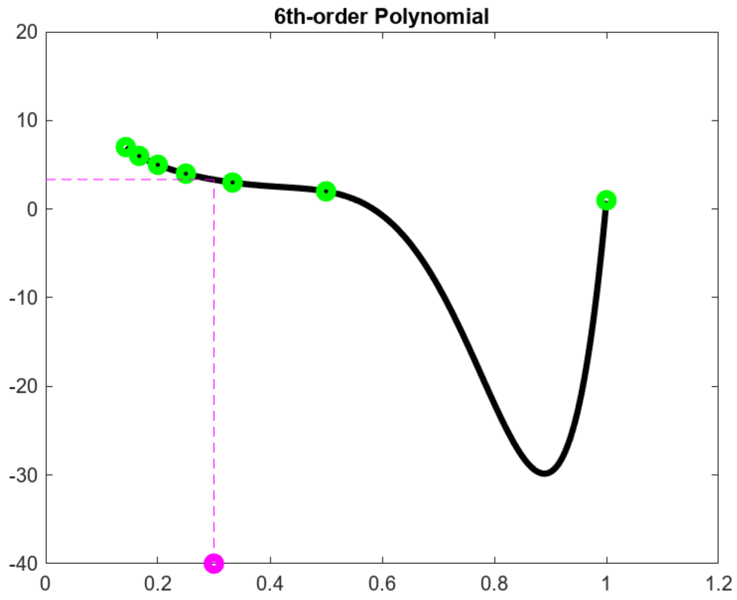
- Let's assume we don't know the real function $f(x) = \frac{1}{x}$, but have the following data

$\boldsymbol{x}$	1	2	3	4	5	6	7
$\boldsymbol{f(x)}$	1	0.5	0.3333	0.25	0.2	0.1667	0.1429



- For the *inverse* interpolation, when $f(x) = 0.3, x = ?$

$\boldsymbol{f(x)}$	0.1429	0.1667	0.2	0.25	0.3333	0.5	1
$\boldsymbol{x}$	7	6	5	4	3	2	1



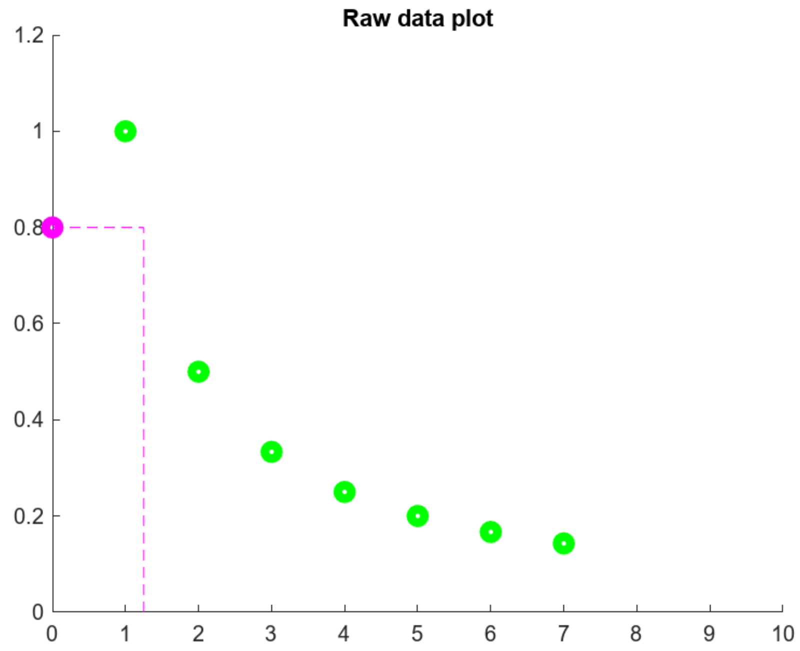
Inverse interpolation

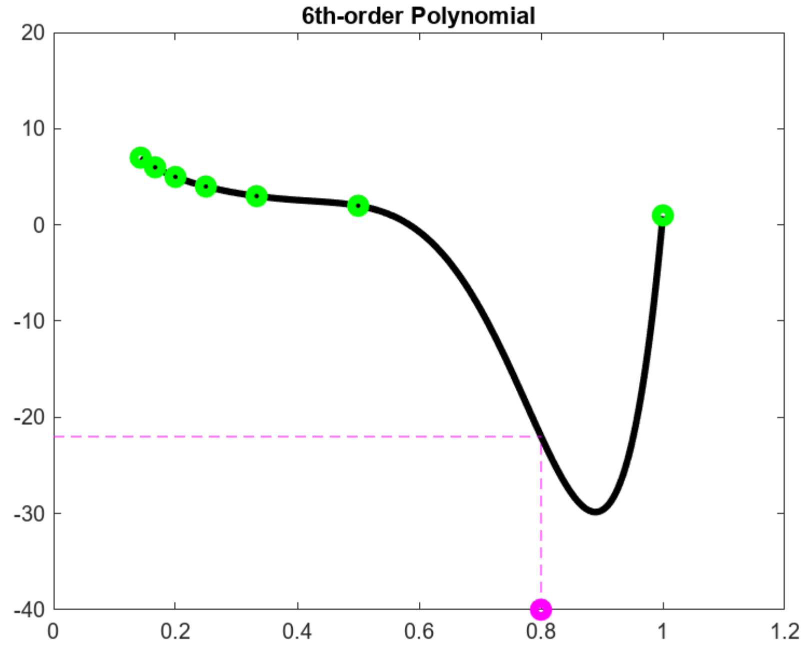
- Let’s assume we don’t know the real function $f(x) = \frac{1}{x}$, but have the following data

$\boldsymbol{x}$	1	2	3	4	5	6	7
$\boldsymbol{f(x)}$	1	0.5	0.3333	0.25	0.2	0.1667	0.1429

- For the *inverse* interpolation, when $f(x) = 0.8$, $x = ?$

$\boldsymbol{f(x)}$	0.1429	0.1667	0.2	0.25	0.3333	0.5	1
$\boldsymbol{x}$	7	6	5	4	3	2	1


Raw data plot


6th-order Polynomial

Inverse interpolation

- When we reverse the variables, there is no guarantee that the values along the new abscissa [$f(x)$'s] will be evenly spaced. The values of even be "telescoped", or have the appearance of a logarithmic scale with some adjacent points bunched together and others spread out widely.

- Such non-uniform spacing on the abscissa often leads to oscillations in the resulting interpolating polynomial.

Inverse interpolation by root finding method

- An alternative strategy is to fit an nth-order interpolating polynomial, $f_n(x)$ to the original data because the x's are evenly spaced, so the resulting polynomial will **not be ill-conditioned**.

- To find the value of x that makes this polynomial equal to the given $f(x)$ then is to solve a roots problem.

Inverse interpolation by root finding of quadratic polynomial

- We can fit a simple quadratic polynomial to the three points: $(2, 0.5), (3,0.3333), (4,0.25)$.

- The result polynomial would be $f_2(x) = 0.041667x^2 - 0.375x + 1.08333$

- Then we can find the roots of $0.3 = 0.041667x^2 - 0.375x + 1.08333$

- $x = \frac{0.375 \pm \sqrt{(-0.375)^2 - 4(0.041667)(1.08333)}}{2(0.041667)} = 5.704158, \quad 3.295842$

- Note that 3.296 is a good approximation of the true value of $\mathbf{3.333}$.

- If we need more accuracy, a third- ,fourth-, fifth-, or sixth-order polynomial along with one of the root-finding methods can be used.

Inverse interpolation by root finding of 6th-order polynomial

- Now for our example, we want to know what value of x when $f(x) = 0.3$.

- We can fit a 6th -order polynomial to the seven points.

- The result polynomial would be

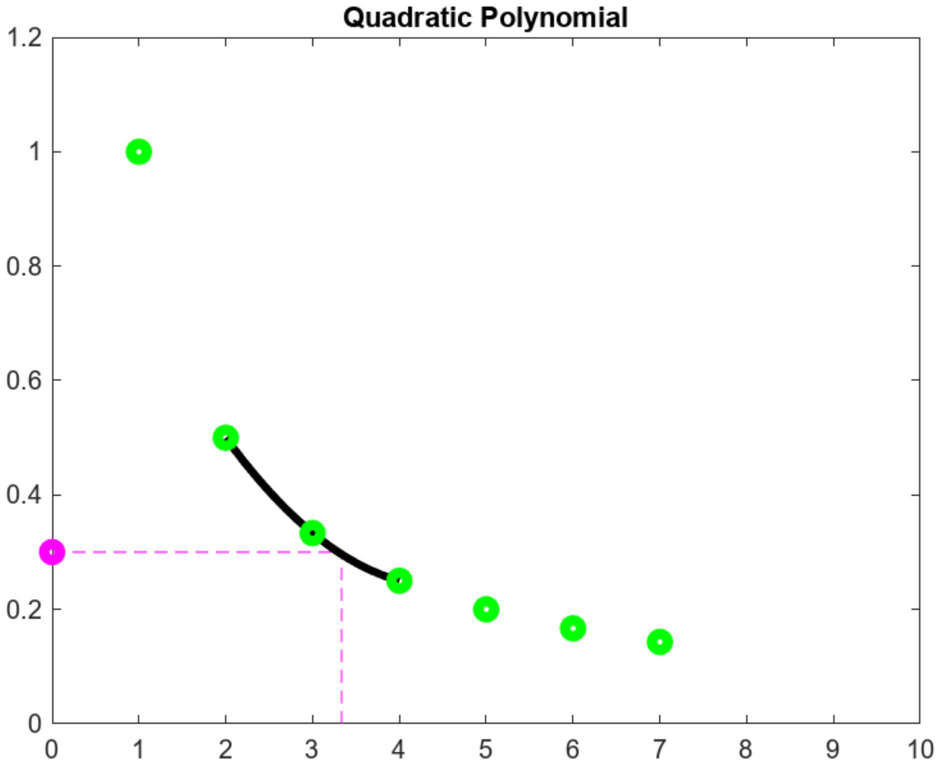
$$f_6(x) = 0.0002x^6 - 0.0055x^5 + 00637x^4 - 0.3879x^3 + 1.3405x^2 - 2.6025x + 2.5915$$

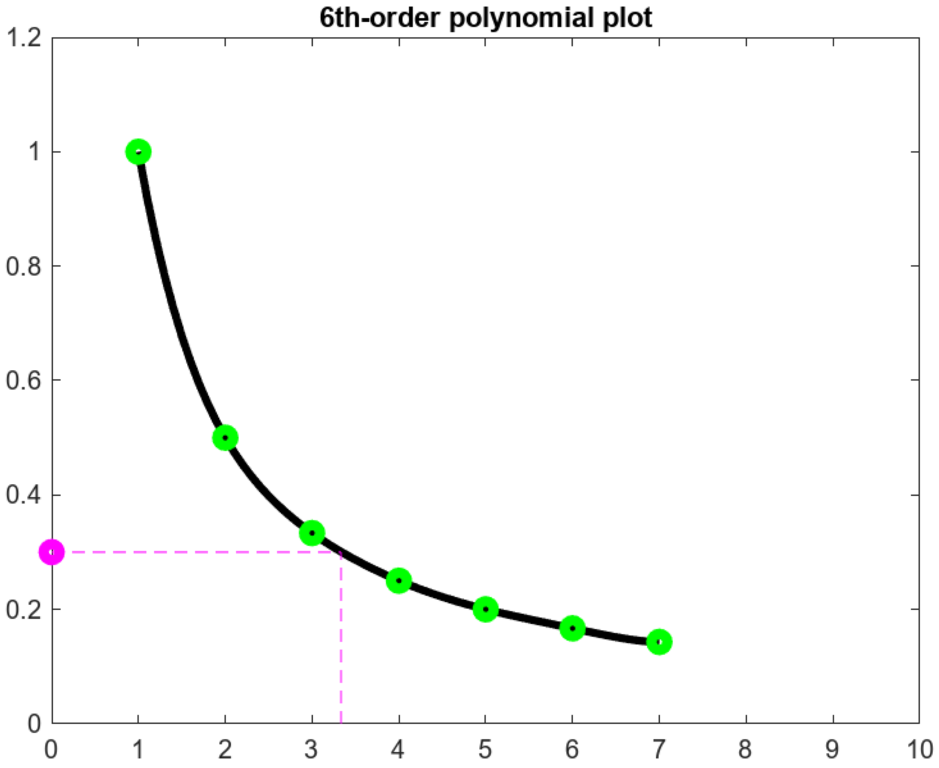
- Then we can find the roots of

$$0.3 = 0.0002x^6 - 0.0055x^5 + 00637x^4 - 0.3879x^3 + 1.3405x^2 - 2.6025x + 2.5915$$

- Even we can use one of the root finding methods, but with the higher order polynomial, finding roots is still not convenient.

Comparison of quadratic and 6th-order polynomial fittings

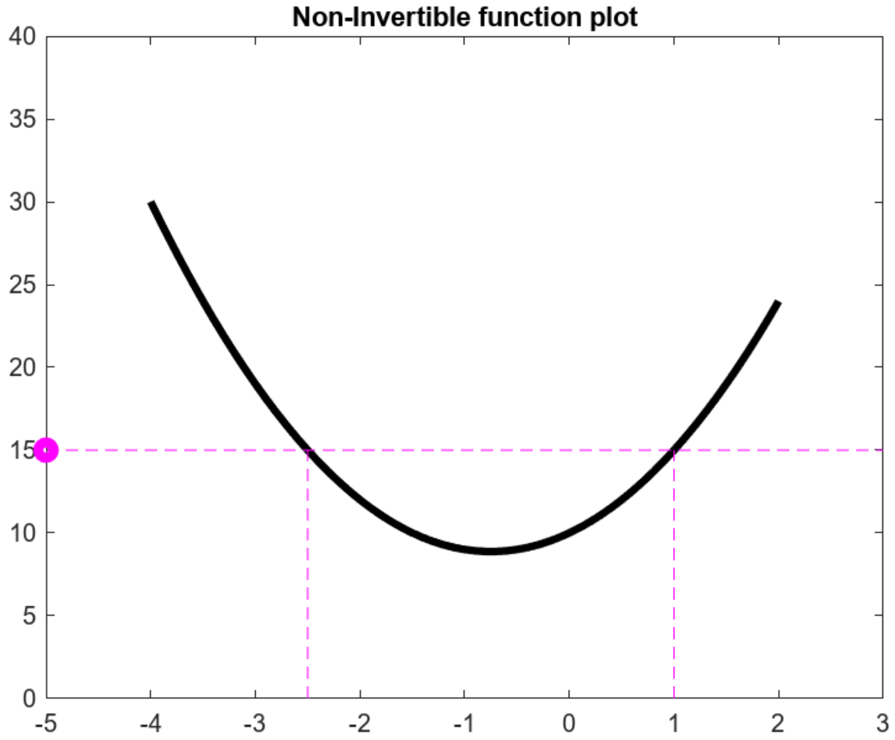




Case 2 : Non-Invertible function

- Let’s deal with the case that the underlying model is a non-invertible function.

- For example, for the invertible function $f(x) = 2x^2 + 3x + 10$, if we want to know what value of x when $f(x) = 15$, then there are more than one value of x.

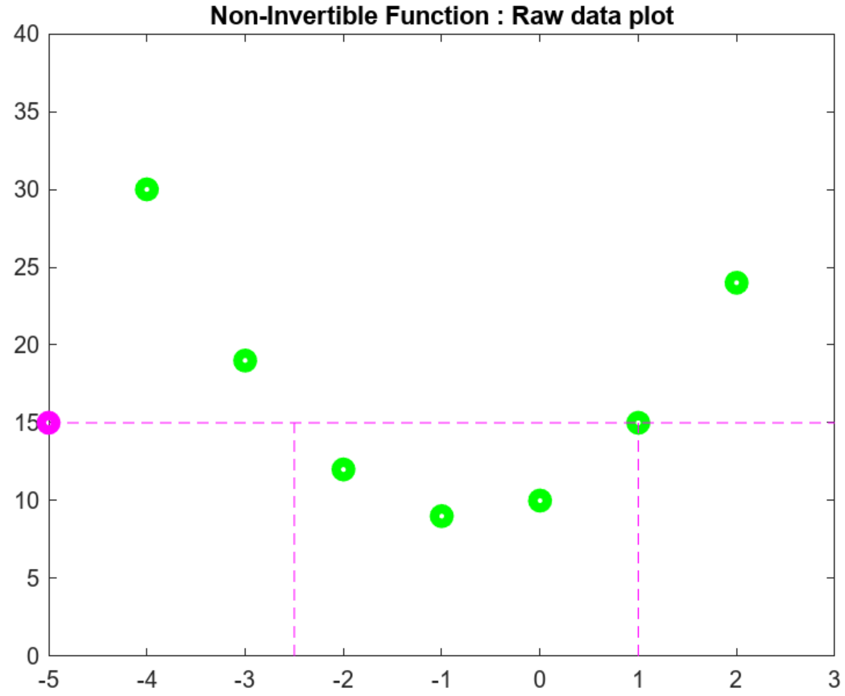


Inverse interpolation

- Let's assume we don't know the real function $f(x) = 2x^2 + 3x + 10$, but have the following data

$\boldsymbol{x}$	-4	-3	-2	-1	0	1	2
$\boldsymbol{f(x)}$	30	19	12	9	10	15	24

- For the *inverse* interpolation, when $f(x) = 15$, $x = ?$

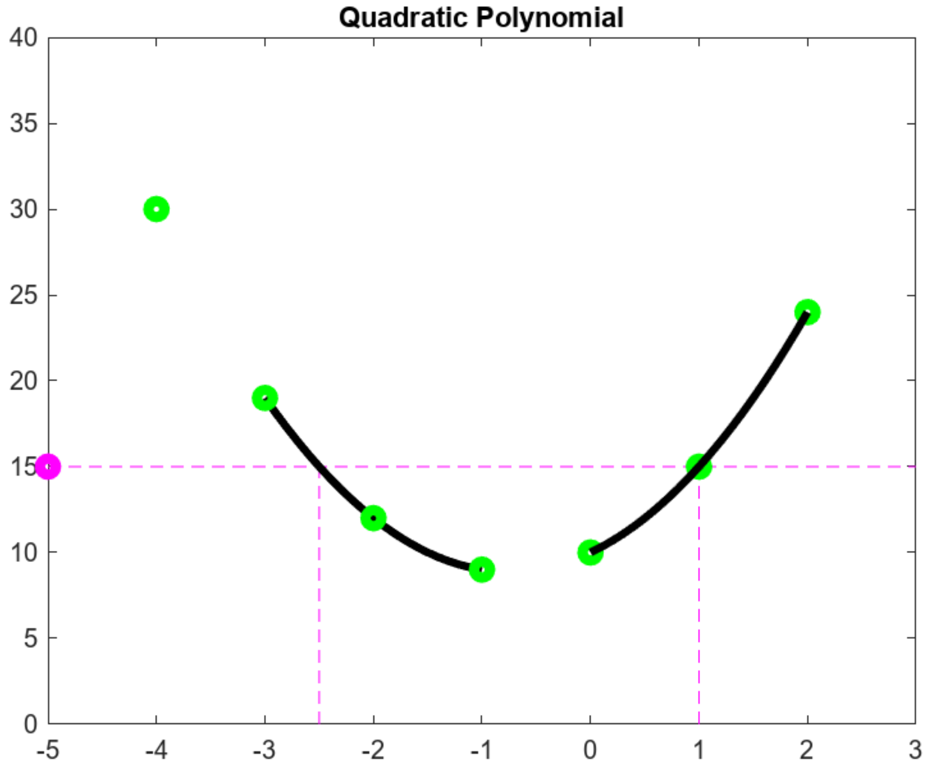

Non-Invertible Function : Raw data plot

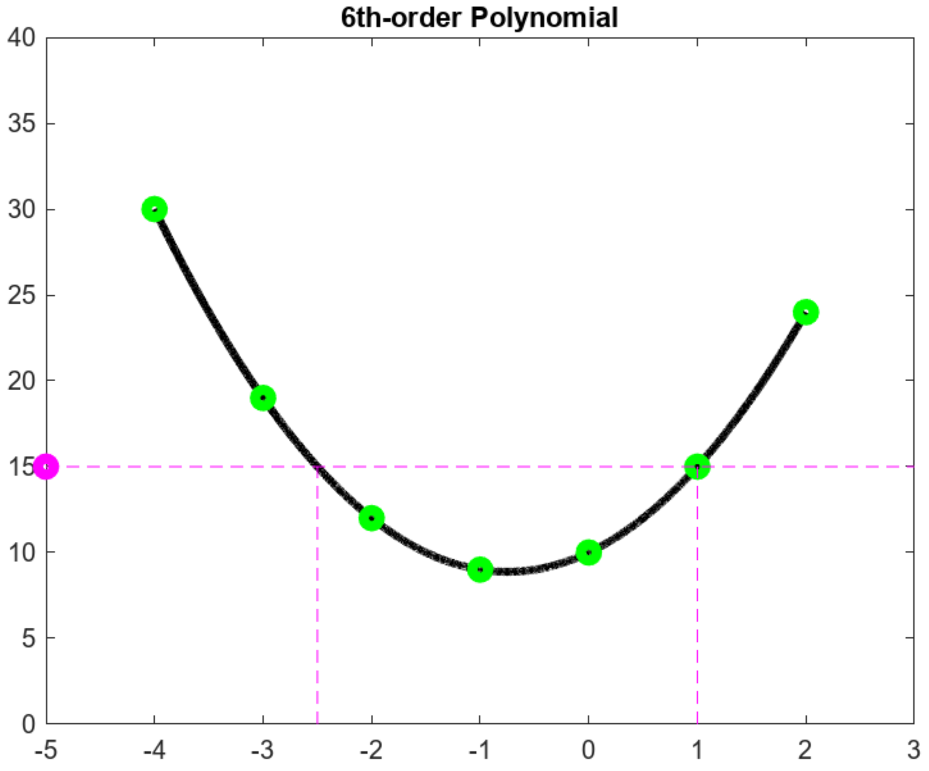
Inverse interpolation by root finding method

- We can fit an nth-order interpolating polynomial, $f_n(x)$ to the original data because the x’s are evenly spaced, so the resulting polynomial will **not be ill-conditioned**.

- To find the value of x that makes this polynomial equal to the given $f(x)$ then is to solve a roots problem.

Inverse interpolation by root finding of quadratic and 6th-order polynomials



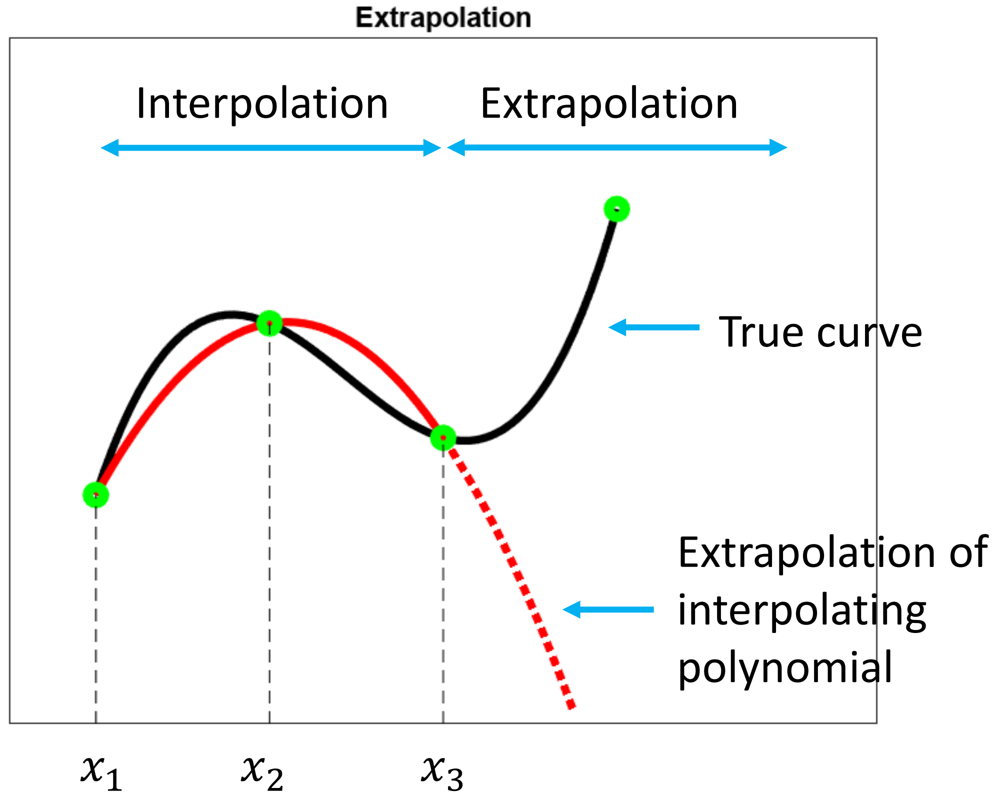


Extrapolation

- Extrapolation : the process of estimating a value of $f(x)$ that lies outside the range of the known base points, $x_1, x_2, \dots, x_n$.

- Extrapolation represents a step into the unknown because the process extends the curve beyond the known region.

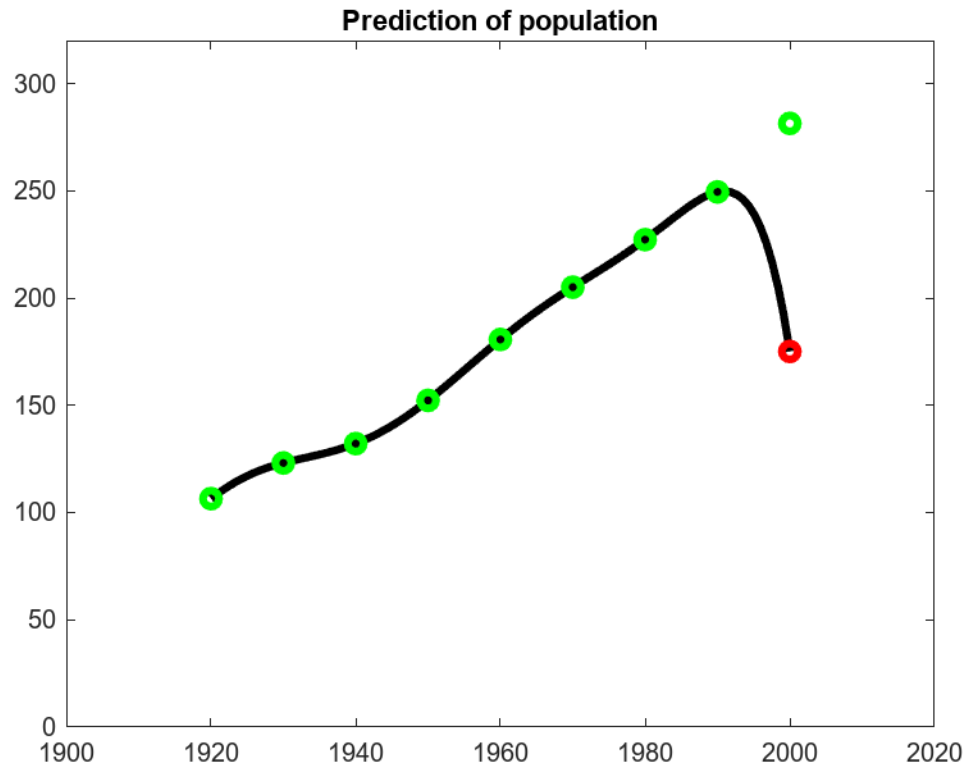
- The true curve can easily diverge from the prediction.



Dangers of Extrapolation

- Population in millions of the USA from 1920 to 2000

- Fit a 7th –order polynomial to the first 8 points (1920-1990) and use it to compute the population in 2000 by extrapolation and compare the prediction with the actual result



Date	1920	1930	1940	1950	1960	1970	1980	1990	2000
Population	106.46	123.08	132.12	152.27	180.67	205.05	227.23	249.46	281.42

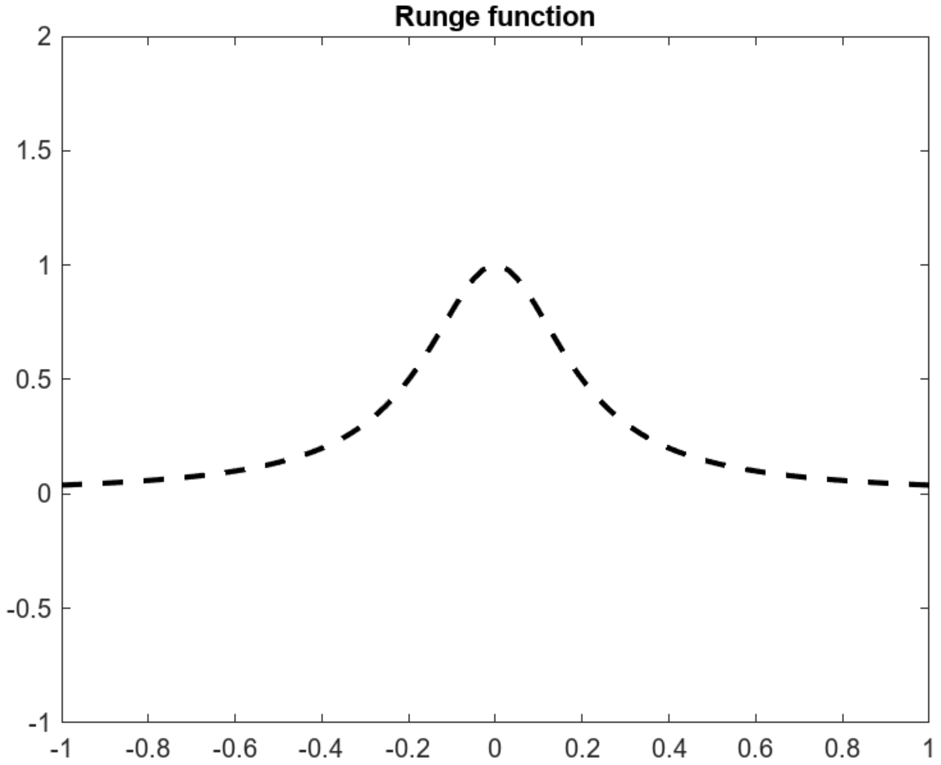
Oscillations

- Higher-order polynomials tend to be very ill-conditioned – they tend to be highly sensitive to round off error

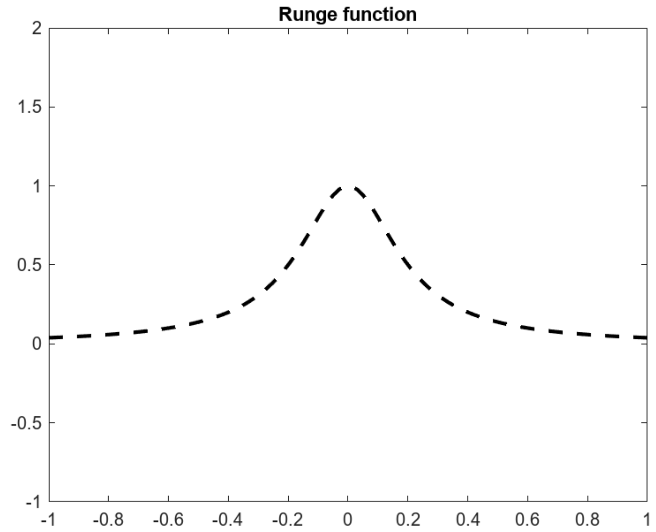
- In 1901, Carl Runge published a study on the dangers of higher-order polynomial interpolation by demonstrating oscillation effects on *Runge's function* :

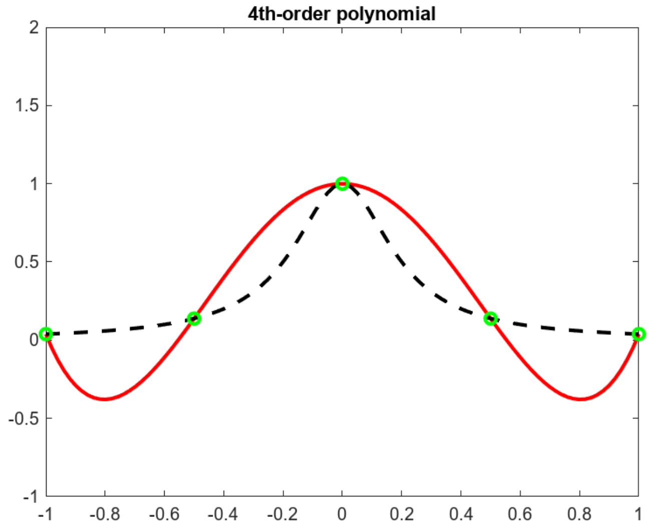
$$f(x) = \frac{1}{1 + 25x^2}$$

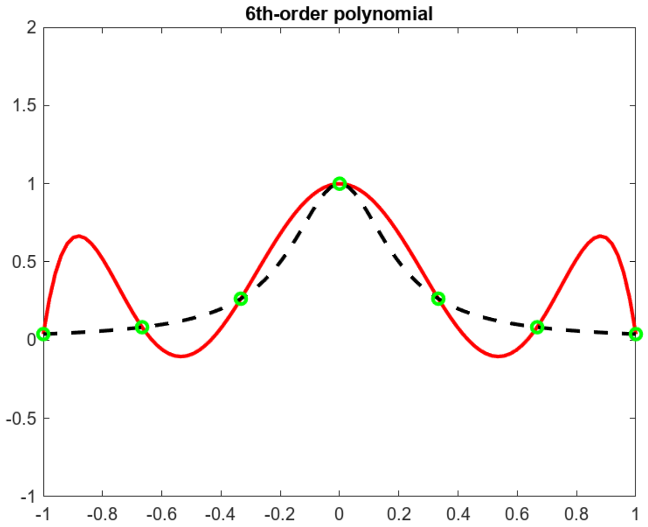
- In case of Runge's function, as we interpolate polynomials of increasing order, the polynomials and the original curve differed considerably.

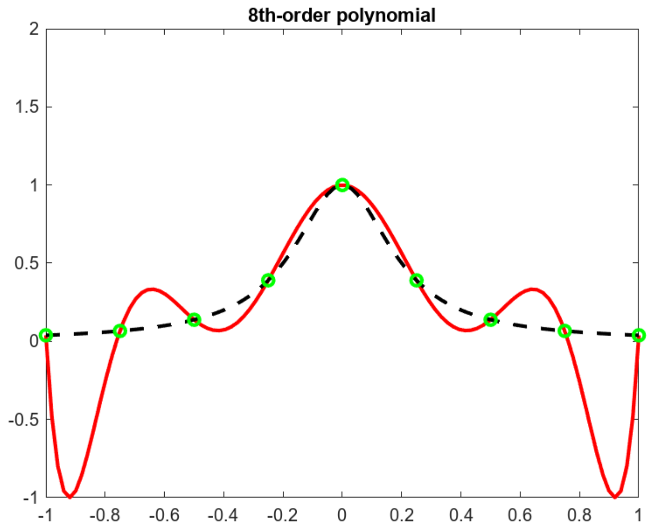


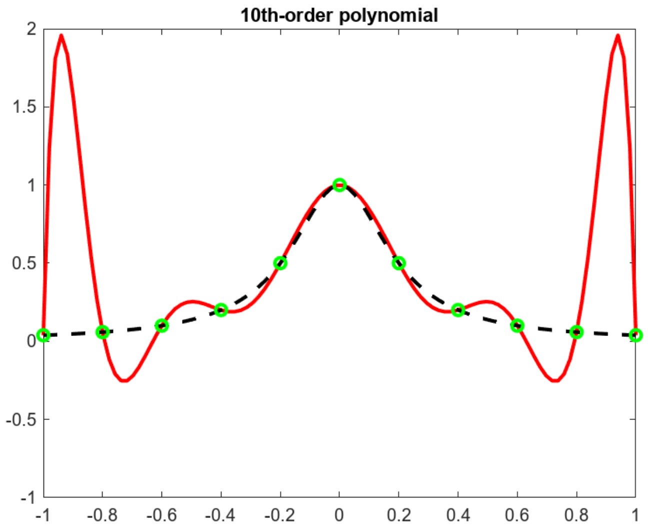
Dangers of higher-order polynomial interpolation


Runge function


4th-order polynomial


6th-order polynomial


8th-order polynomial


10th-order polynomial

About higher-order polynomials

- Although there may be certain contexts where higher-order polynomials are necessary, they are usually to be avoided.

- Lower-order polynomials can be used effectively to capture the curving trends of data without suffering from oscillations.