# ENGSCI 233 – Computational Techniques and Computer Systems

## Week 3: Sampled Data (Interpolation)

Andreas W. Kempa-Liehr
based on material from David Dempsey and Colin Simpson


Department of Engineering Science

Week 3 term 1223 – slides˙sampleddata.pdf, Rev. 2c3afd4

# Sections

# Introduction

This topic considers algorithmic implementation of:

- Interpolation - estimate the value of some process using currently available data.
- Integration - numerical estimation of an integral.

# Learning Outcomes – Interpolation

- Develop algorithms to apply concepts of numerical interpolation (e.g. polynomial fitting, linear interpolation and cubic splines) and numerical integration (e.g. Newton-Cotes and Gaussian quadrature methods) to discrete data.
- Know the pros and cons of linear, polynomial and spline interpolation. Know how these may extend to extrapolation.

# Sampled Data

Sampled data: a set of measurements of a continuous process (e.g. velocity, force) made at discrete points (e.g. time, position).

Terminology used throughout this topic:

- The independent variable (e.g. time) is $x$.
- The continuous process, or dependent variable, (e.g. temperature) is $y(x)$.
- The set of $n$ measurement points are $[x_0, x_1, \cdots x_{n-1}]$ or, more compactly, $x_i$.
- The set of $n$ measurements values are $[y(x_0), y(x_1), \cdots y(x_{n-1})]$, or $[y_0, y_1, \cdots y_{n-1}]$, or $y_i$.

# Example Data

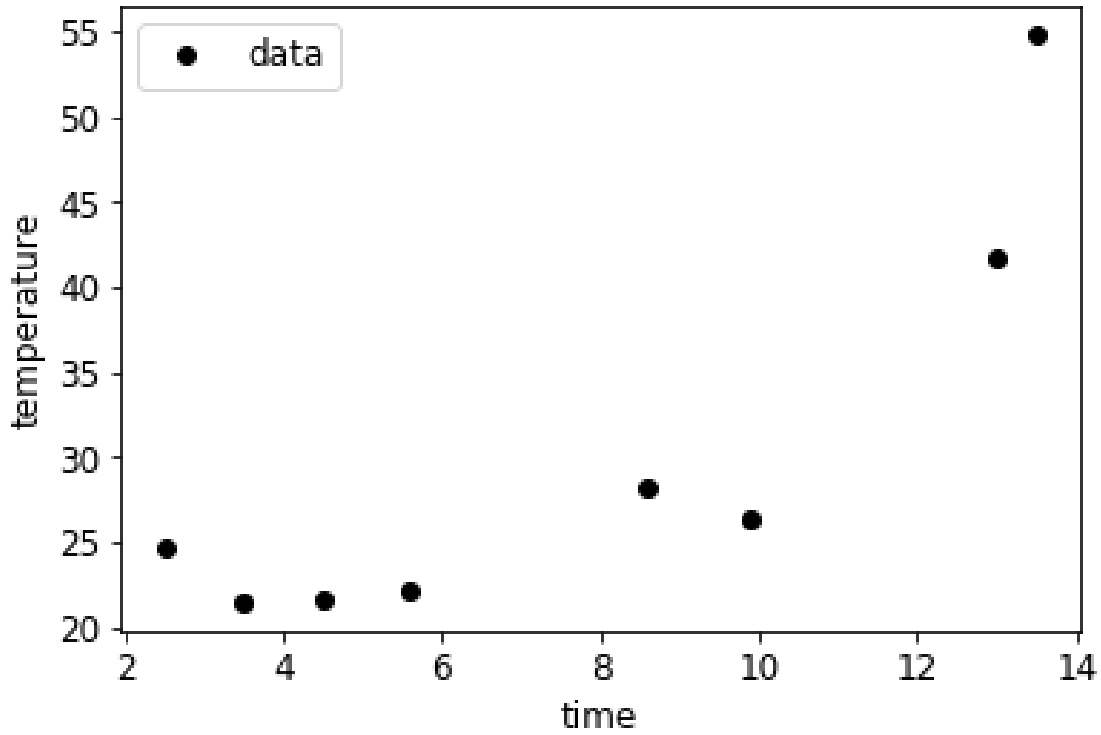
For example, a set of eight measurements of temperature:

| Time | 2.5 | 3.5 | 4.5 | 5.6 | 8.6 | 9.9 | 13.0 | 13.5 |
|---|---|---|---|---|---|---|---|---|
| Temp | 24.7 | 21.5 | 21.6 | 22.2 | 28.2 | 26.3 | 41.7 | 54.8 |

- The independent variable is time, $t$.
- The dependent variable is temperature, $T$.
- The set of $n = 8$ measurement points are $t_i$.
- The set of $n = 8$ measurement values are $T_i = T(t_i)$.

# Data Visualisation

A basic visualisation of the example sampled data:




The sampled data represent an incomplete picture of the continuous process, $T(t)$.

# Introduction to Interpolation

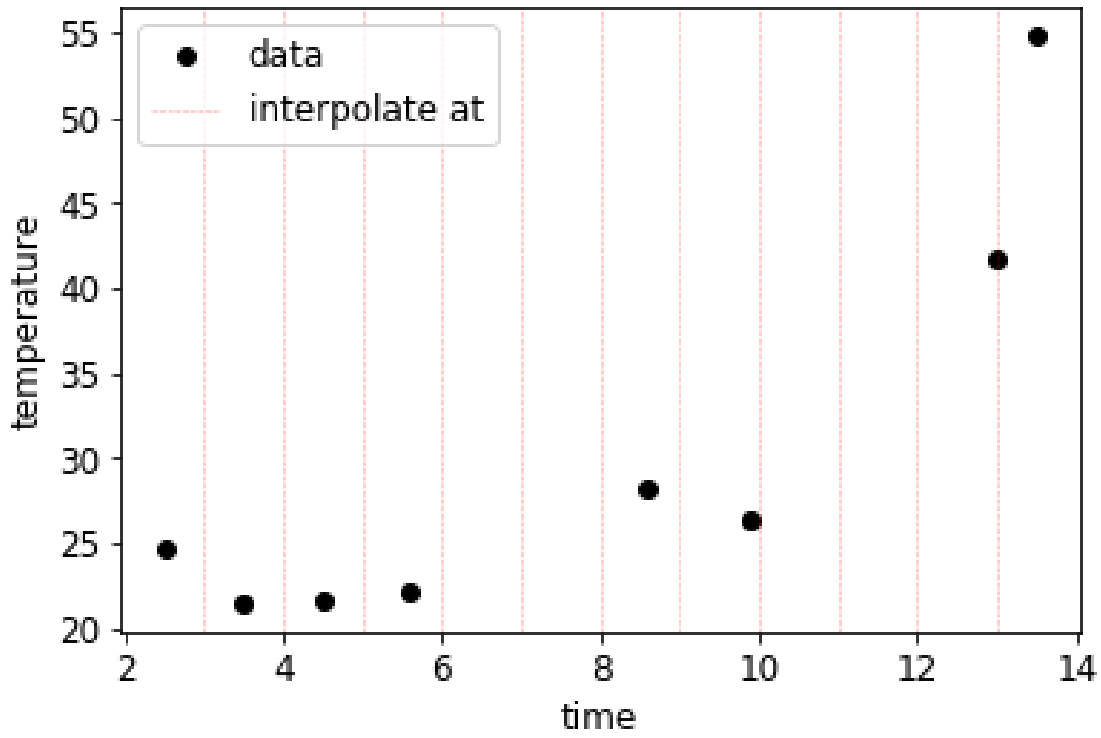
Often we require an estimate of the dependent variable at a point where no data are available. For example, what is $T(7)$?

Unknown measurements will be referred to as $y_j = y(x_j)$, whereas the available data will be referred to as $y_i = y(x_i)$.

Interpolation allows us to approximate these $y_j$ at measurement points that are within the current data.

# Interpolation Points

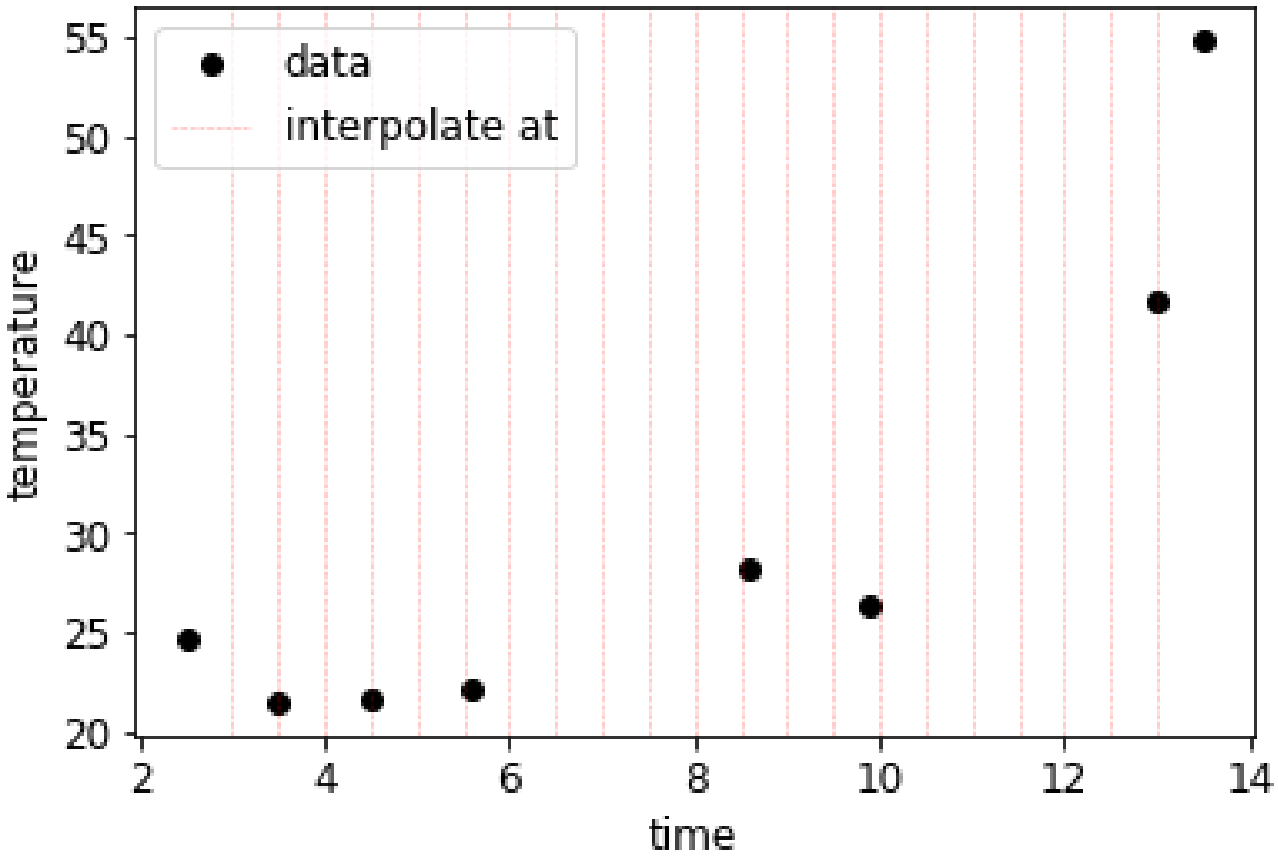
One possible set of interpolation points, with an interval of $t = 1$:




These points must be within the bounds of the existing data.

# Interpolation Points

Another set of interpolation points, with an interval of $t = 0.5$:



# Interpolation Methods

All of the interpolation methods covered in this course can be divided into two steps:

1. Find an interpolating function, $f(x_i)$, that exactly or approximately matches the data, $y(x_i)$.
2. Evaluate the interpolating function, $f(x)$, at the points of interest, $f(x_j) = y(x_j)$.

# Extrapolation

If we wish to estimate the dependent variable outside of the current data, this process is alternatively referred to as **extrapolation** rather than interpolation.

An extrapolating function, $g(x)$, is required to evaluate the extrapolated points. However, extrapolation has inherent issues with accuracy as it is not bounded by any known data.

We do not cover specific extrapolation methods in this course.

# Introduction to Polynomial Fitting

Fit a single interpolating function, $f(x)$, to all of the sampled data.

This function can take the form of an order $m$ polynomial:

$$f(x) = a_0 + a_1 x + a_2 x^2 + \ldots a_m x^m$$

Or in *summation notation*:

$$f(x) = \sum_{k=0}^{m} a_k x^k$$

# Polynomial Order

Consider a set of $n$ measurement values - there exists a unique polynomial of order $\leq (n-1)$ that will exactly match the data.

Would this make a suitable interpolating function?

There are two obvious cases where it wouldn't be suitable:

- Sampled data are noisy due to measurement error.
- Large number of data points.

# Quantifying Best Fit

In general, we aim to find the lowest order polynomial that fits the data 'reasonably well'. We can use the Residual Sum of Squares, RSS, to quantify 'goodness of fit':

$$\text{RSS} = \sum_{i=0}^{n-1} \left(y\left(x_i\right) - f\left(x_i\right)\right)^2$$

$$\text{RSS} = \sum_{i=0}^{n-1} \left(y\left(x_i\right) - \left(a_0 + a_1 x_i + a_2 {x_i}^2 + \ldots + a_m {x_i}^m\right)\right)^2$$

where we find the polynomial coefficients that minimise RSS.

**Note:** RSS is not the only way of quantifying 'goodness of fit'.

# Minimising RSS

How do we minimise RSS when fitting a polynomial of order $m$?

ENGSCI 211 covered a method for finding the stationary points of a function of multiple independent variables:

$$\frac{\partial\,(\text{RSS})}{\partial a_0} = \frac{\partial\,(\text{RSS})}{\partial a_1} = \frac{\partial\,(\text{RSS})}{\partial a_2} = \ldots = \frac{\partial\,(\text{RSS})}{\partial a_m} = 0$$

We require $m+1$ derivative equations to solve for the $m+1$ polynomial coefficients.

# Polynomial Coefficients

These $m+1$ equations can be written in a matrix form, $A\vec{x} = \vec{b}$:

$$\begin{bmatrix} n & \sum x_i & \sum x_i^2 & \cdots & \sum x_i^m \\ \sum x_i & \sum x_i^2 & \sum x_i^3 & \cdots & \sum x_i^{m+1} \\ \sum x_i^2 & \sum x_i^3 & \sum x_i^4 & \cdots & \sum x_i^{m+2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \sum x_i^m & \sum x_i^{m+1} & \sum x_i^{m+2} & \cdots & \sum x_i^{2m} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_m \end{bmatrix} = \begin{bmatrix} \sum y_i \\ \sum x_i y_i \\ \sum x_i^2 y_i \\ \vdots \\ \sum x_i^m y_i \end{bmatrix}$$

- $A$ is known as the Vandermonde matrix.
- $\vec{x}$ can be solved for the polynomial coefficients.
- $\vec{b}$ includes terms from the sampled data.

# Algorithm

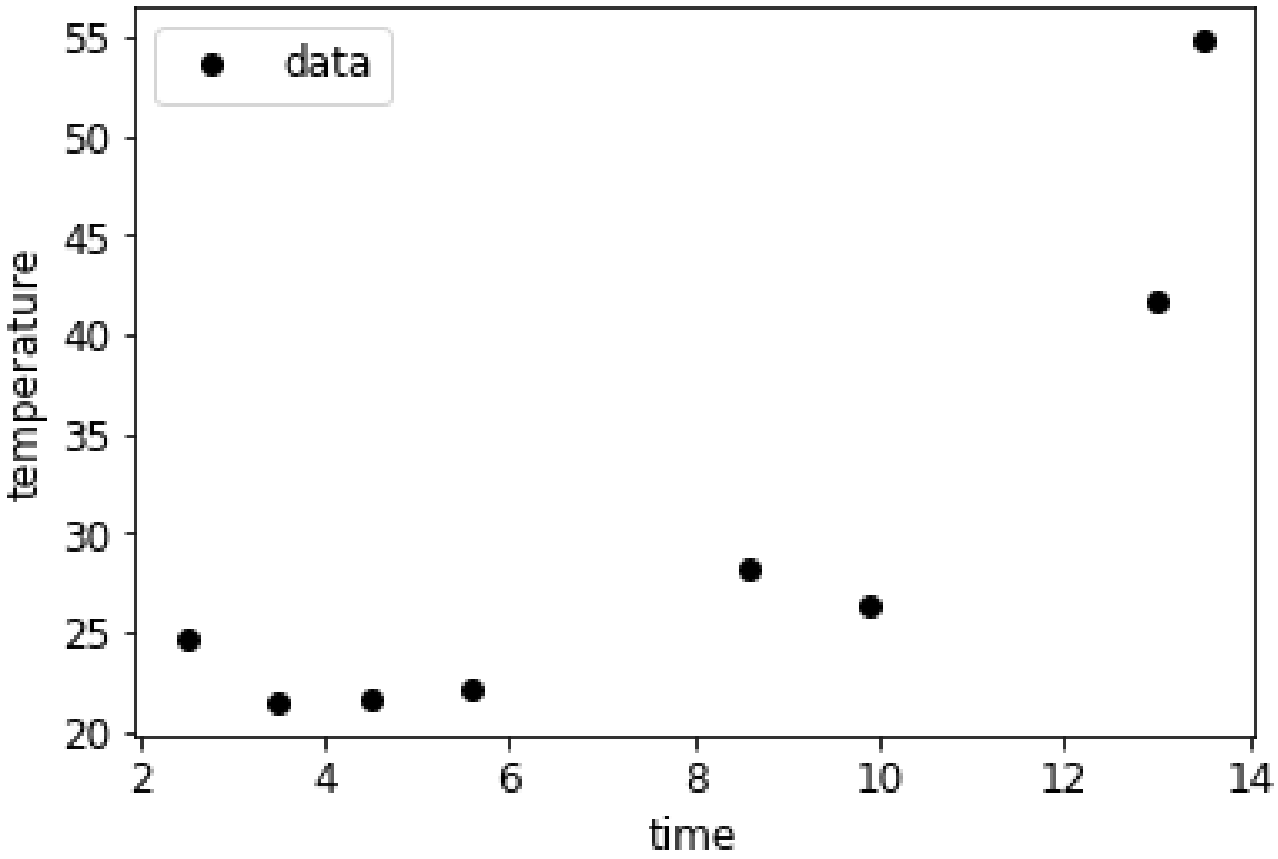
Polynomial fitting can be solved as a linear algebra problem:

- Initialise the Vandermonde/coefficient matrix, $A$
- Initialise the vector of constants, $\vec{b}$
- Solve $A\vec{x} = \vec{b}$ using a linear algebra solver.

Once the polynomial coefficients, contained within $\vec{x}$, have been found, the polynomial can be evaluated at the interpolation points.

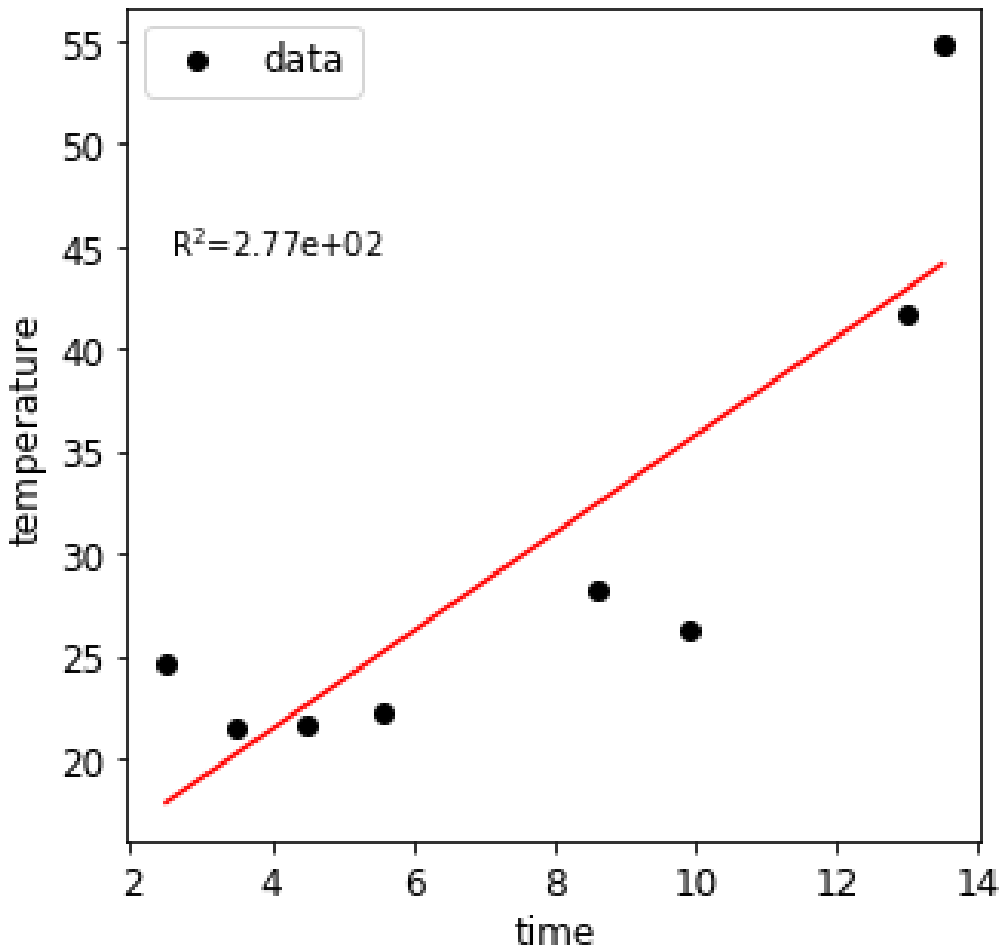
# Sampled Data

Consider again the sampled data example from before:



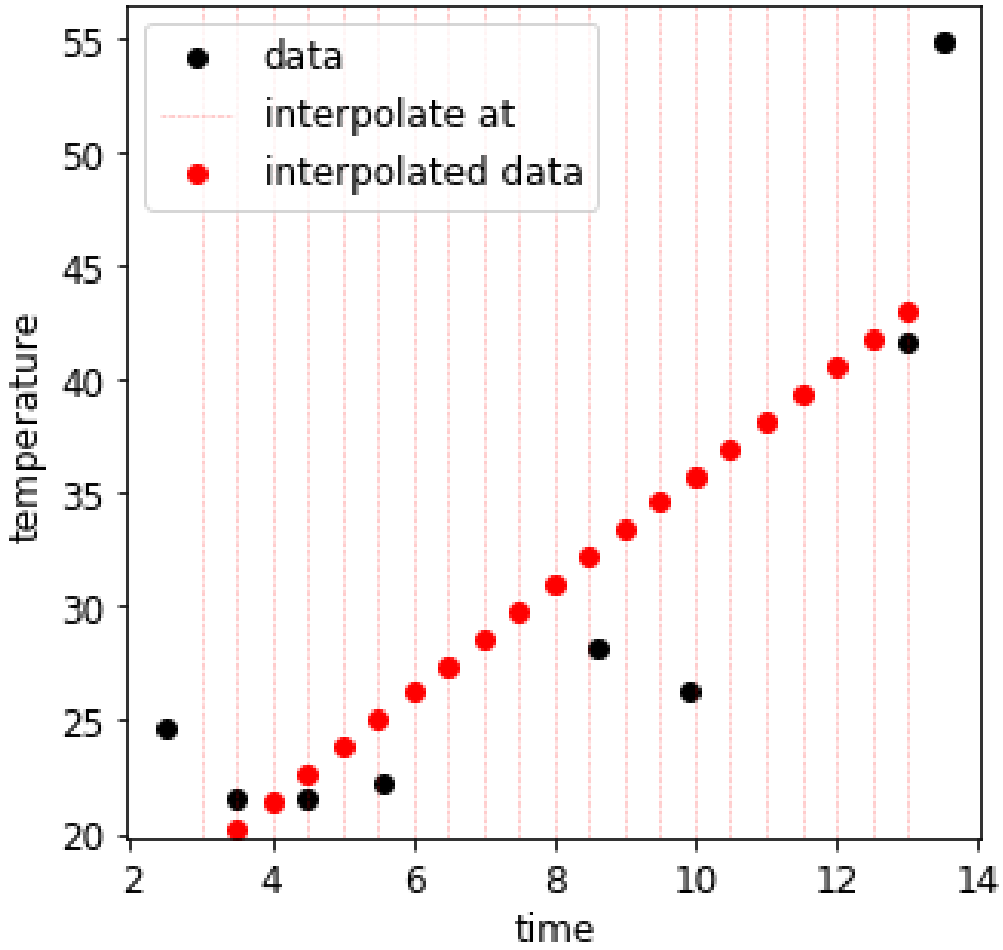
# Order 1 Polynomial

Fitting an order 1 best-fit polynomial (i.e. straight line) to the data, with an associated RSS $\approx 277$:



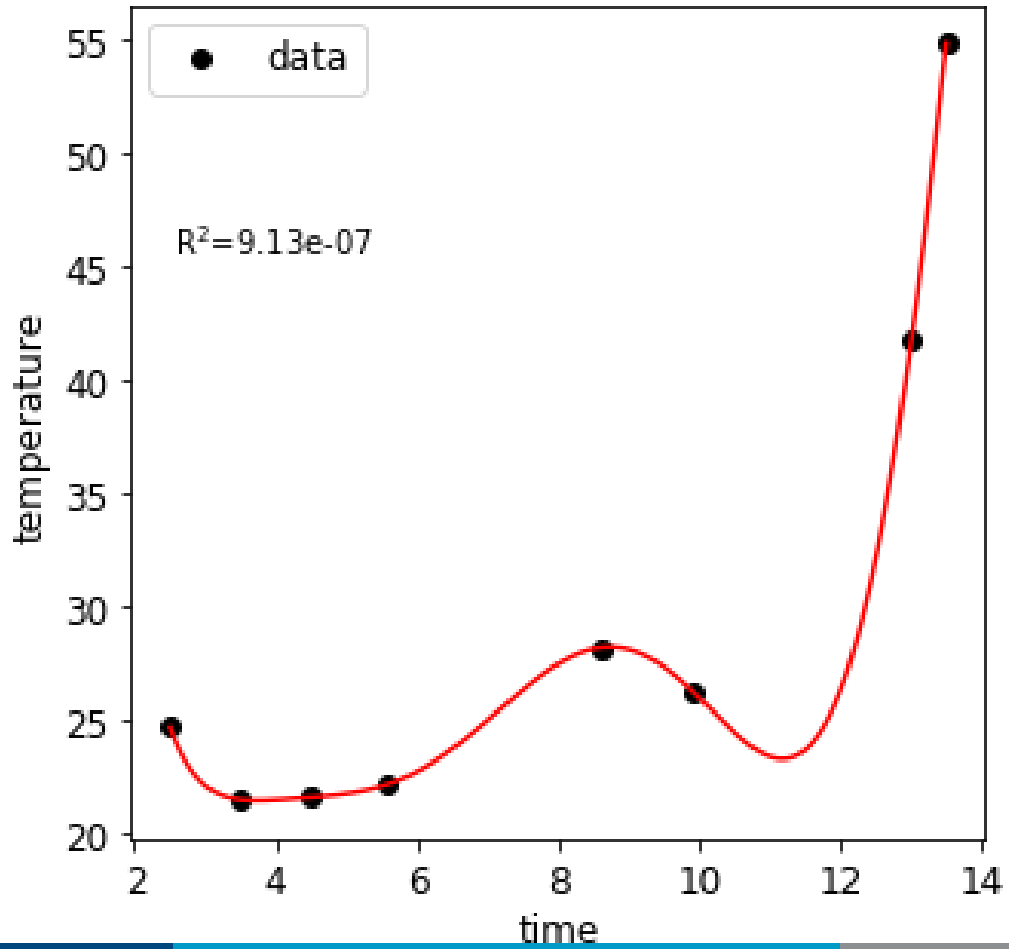
# Interpolated Points

Evaluating the fitted polynomial at the interpolation points:



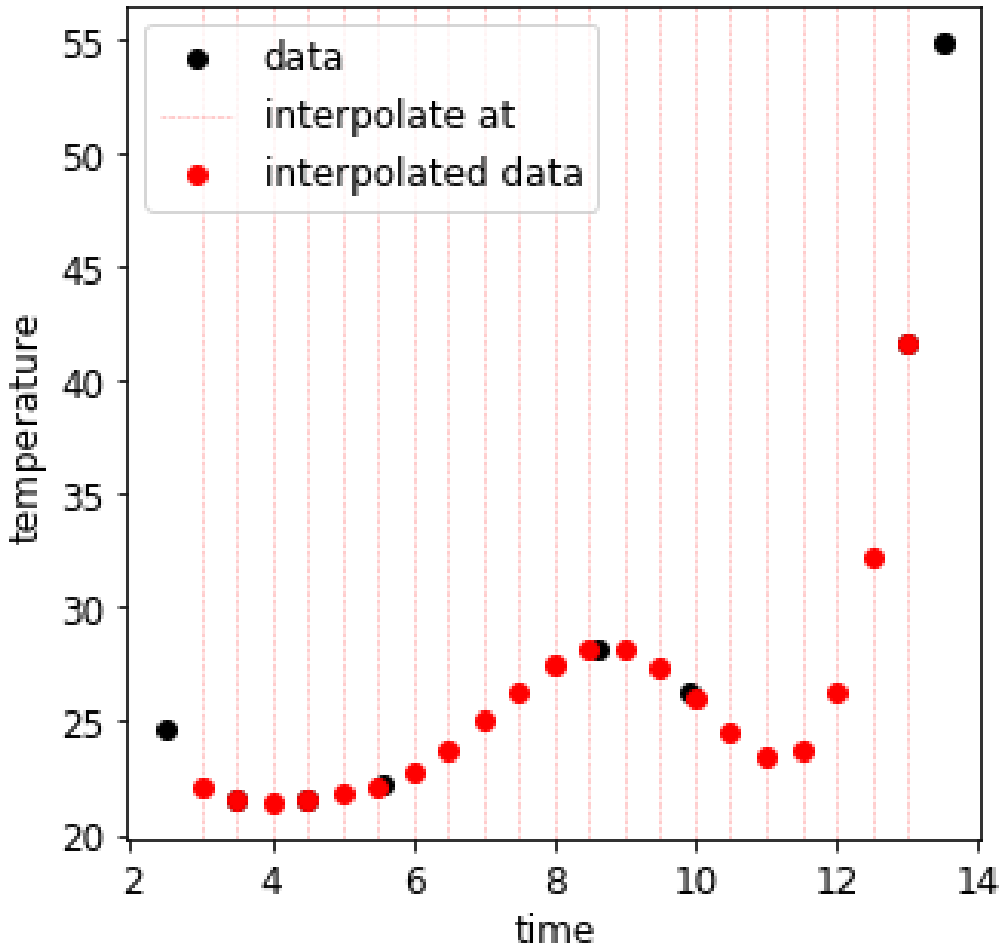
# Order 7 Polynomial

For $n = 8$ data points, there exists an order $m = n - 1$ polynomial that exactly matches the data (i.e. RSS $= 0$):



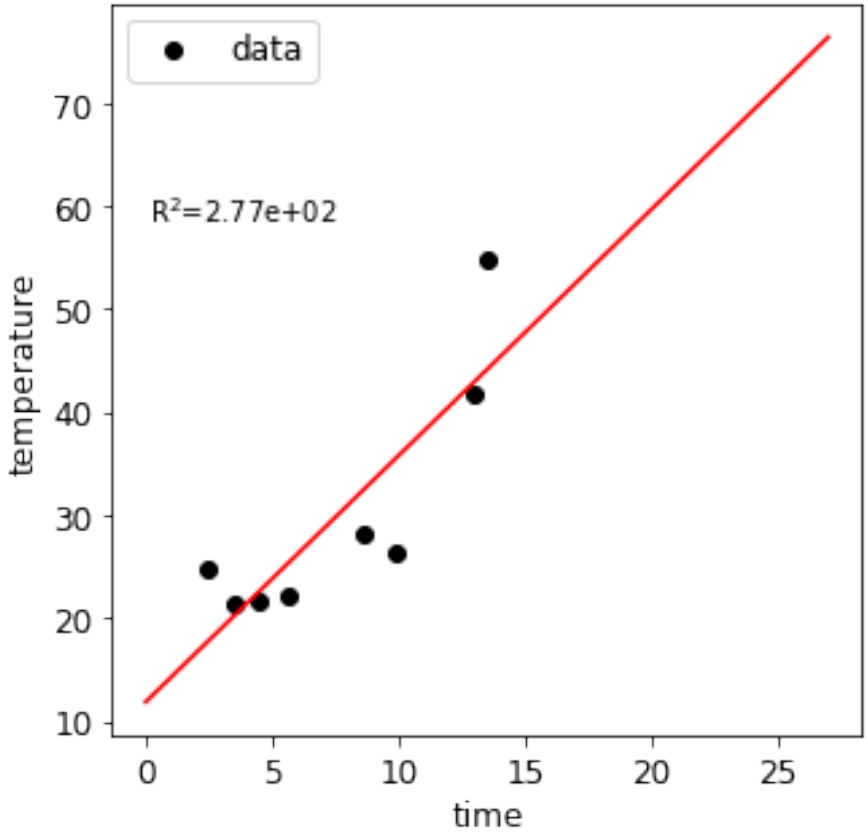
# Interpolated Points

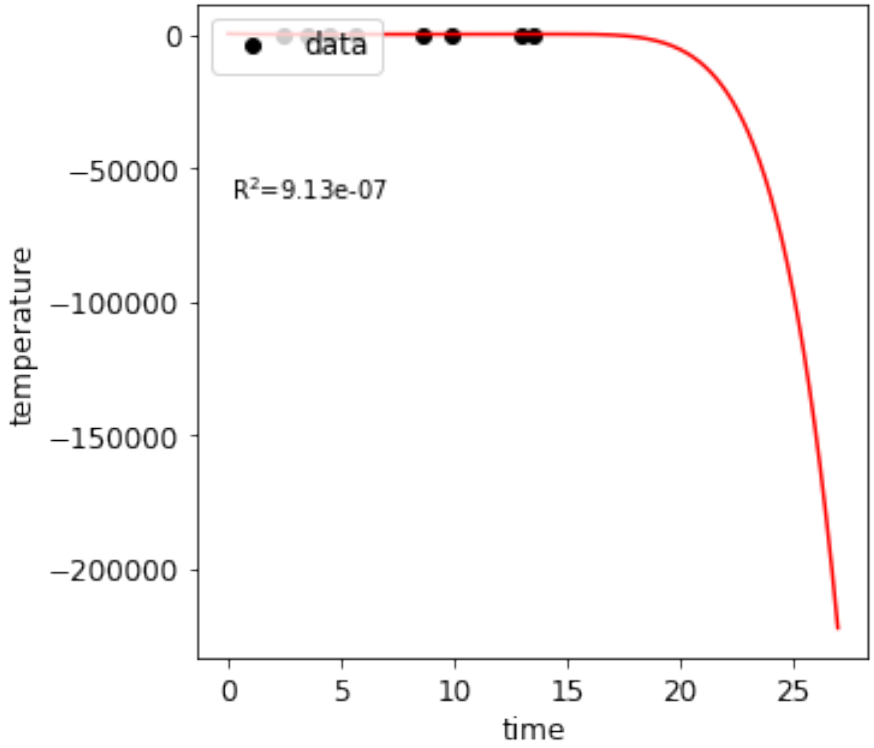
Evaluating this fitted polynomial gives a very different interpolation to the order 1 polynomial:



# Extrapolation

Consider extrapolating with order 1 and 7 polynomials:



# Introduction

Rather than fit one interpolating function to $n$ measurements, we can fit $n-1$ functions, $f_i$, to each pair of measurements:

- $f_0$ is an interpolating function in domain $(x_0, x_1)$
- $f_1$ is an interpolating function in domain $(x_1, x_2)$
- ...
- $f_{n-2}$ is an interpolating function in domain $(x_{n-2}, x_{n-1})$

# Straight Line Equation

Piecewise linear uses straight lines as interpolating functions for each measurement pair.

Consider fitting a straight line between $y(x_0)$ and $y(x_1)$:

$$f_0(x) = m_0 x + c_0$$

$$f_0(x) = \underbrace{\frac{y_1 - y_0}{x_1 - x_0}}_{m_0} x + \underbrace{y_0 - \frac{y_1 - y_0}{x_1 - x_0} x_0}_{c_0}$$

$$f_0(x) = y_0 + \frac{y_1 - y_0}{x_1 - x_0}(x - x_0)$$

# Straight Line Diagram

# Subintervals

If there are $n$ measurements in the sampled data, we can divide up the sampled data into $n-1$ subintervals.

The $i^{th}$ subinterval is bounded by the points $[x_i, x_{i+1}]$. We can fit a straight line equation, $f_i(x)$, for each subinterval:

$$f_i(x) = y_i + \frac{y_{i+1} - y_i}{x_{i+1} - x_i}(x - x_i)$$

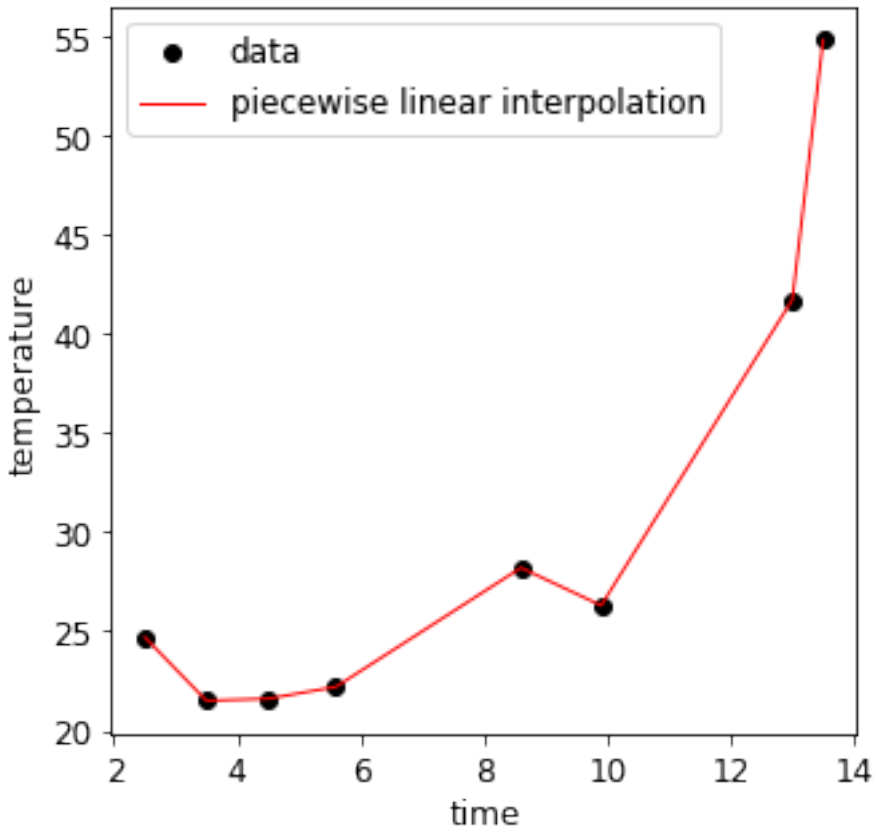
# Algorithm

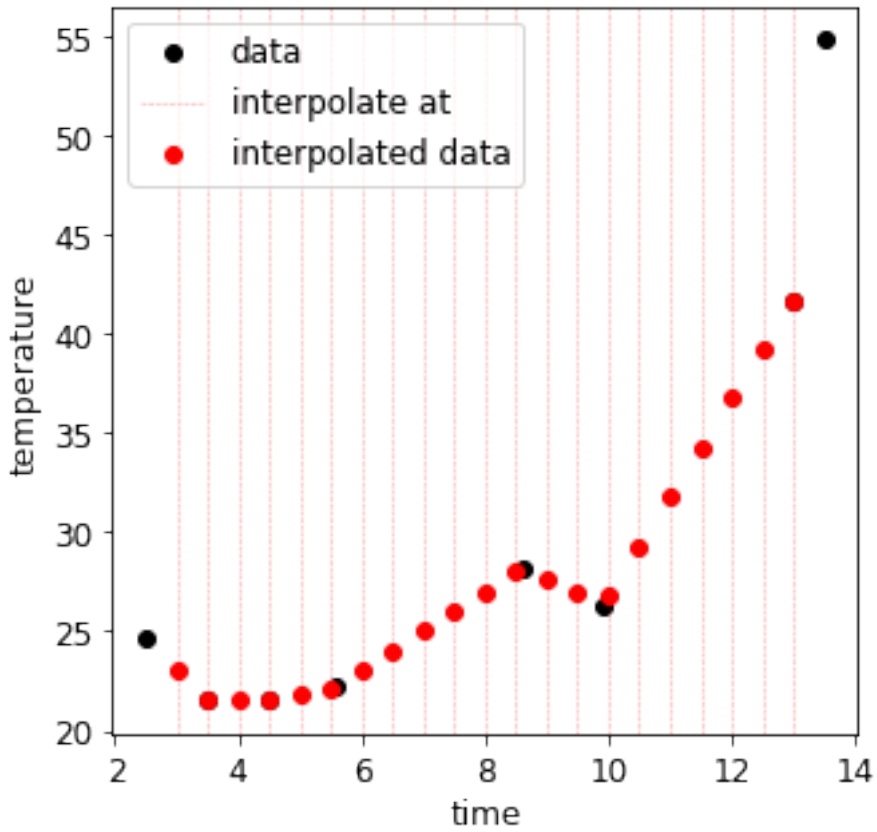
The algorithm steps for implementing piecewise linear interpolation:

- Compute the gradient, $m_i$, and intercept, $c_i$, for each subinterval.
- Find which subinterval each interpolation point, $x_j$, is in.
- Evaluate the relevant interpolating function, $f_i$, at the interpolation point, $f_i(x_j)$.

# Example - Interpolating Functions

The interpolating functions and interpolated points:




Note: this method suffers from derivative discontinuity.

# Introduction

Cubic splines fit an order 3 polynomial to each adjacent data pair. For $n$ measurements, there are $n-1$ subintervals and therefore also $n-1$ cubic polynomials to fit.

Cubic splines have first and second order derivative continuity at the subinterval boundaries. This provides a smoother fit through data points than linear piecewise interpolation.

# Cubic Spline Equation

For subinterval $i$, which covers the interval $[x_i, x_{i+1}]$:

$$p_i(x) = a_0^{(i)} + a_1^{(i)}(x - x_i) + a_2^{(i)}(x - x_i)^2 + a_3^{(i)}(x - x_i)^3$$

The notation used in the Jupyter Notebook:

- $p_i(x)$ is the cubic spline for subinterval $i$
- $a_0^{(i)}$ to $a_3^{(i)}$ are the polynomial coefficients for subinterval $i$

# Derivative Equations

Adjacent cubic splines, $p_i(x)$ and $p_{i+1}(x)$, must have first and second order derivative continuity at $x = x_{i+1}$.

The general equations for the first and second order derivatives:

$$\frac{dp_i}{dx} = a_1^{(i)} + 2a_2^{(i)}(x - x_i) + 3a_3^{(i)}(x - x_i)^2$$

$$\frac{d^2p_i}{dx^2} = 2a_2^{(i)} + 6a_3^{(i)}(x - x_i)$$

# Polynomial Coefficients

For $n$ data points, there are $n-1$ cubic splines and $4\,(n-1)$ unknown polynomial coefficients.

We therefore require $4\,(n-1)$ equations:

- Boundary conditions on function value
- Boundary conditions on function first order derivative
- Boundary conditions on function second order derivative
- Second derivative zero at first/last data points

# Conditions: Function Value

Each cubic spline must exactly match the sampled data at either end of its subinterval i.e. $p_i(x_i) = y(x_i)$ and $p_i(x_{i+1}) = y(x_{i+1})$:

$$a_0^{(i)} = y_i$$

$$a_0^{(i)} + a_1^{(i)}(\Delta x_i) + a_2^{(i)}(\Delta x_i)^2 + a_3^{(i)}(\Delta x_i)^3 = y_{i+1}$$

where $\Delta x_i = x_{i+1} - x_i$.

These boundary conditions apply to all $n-1$ cubic polynomials. This gives $2(n-1) = 2n-2$ equations.

# Conditions: First Order Derivative Continuity

Equating the first order derivative of adjacent cubic splines:

$$\left.\frac{dp_i}{dx}\right|_{x=x_{i+1}} - \left.\frac{dp_{i+1}}{dx}\right|_{x=x_{i+1}} = 0$$

$$a_1^{(i)} + 2a_2^{(i)}\left(\Delta x_i\right) + 3a_3^{(i)}\left(\Delta x_i\right)^2 - a_1^{(i+1)} = 0$$

where $\Delta x_i = x_{i+1} - x_i$.

There are $n-2$ adjacent cubic splines, giving $n-2$ equations.

# Conditions: Second Order Derivative Continuity

Equating the second order derivative of adjacent cubic splines:

$$\left.\frac{d^2 p_i}{dx^2}\right|_{x=x_{i+1}} - \left.\frac{d^2 p_{i+1}}{dx^2}\right|_{x=x_{i+1}} = 0$$

$$2a_2^{(i)} + 6a_3^{(i)}(\Delta x_i) - 2a_2^{(i+1)} = 0$$

where $\Delta x_i = x_{i+1} - x_i$.

There are $n-2$ adjacent cubic splines, giving $n-2$ equations.

# Conditions: End Points

The second derivative is zero at the first data point:

$$\left.\frac{d^2 p_0}{dx^2}\right|_{x=x_0} = 0$$

$$2a_2^{(0)} = 0$$

The second derivative is zero at the last data point:

$$\left.\frac{d^2 p_{n-2}}{dx^2}\right|_{x=x_{n-1}} = 0$$

$$2a_2^{(n-2)} + 6a_3^{(n-2)}\left(\Delta x_{n-2}\right) = 0$$

This gives the last two equations.

# System of Equations

This system of $4\,(n-1)$ equations can be solved as a linear algebra problem i.e. $A\vec{x} = \vec{b}$.

In this course we construct the rows of $A$ and $\vec{b}$ using:

- First $2\,(n-1)$ rows from function value condition.
- Next $n-2$ rows from first derivative continuity.
- Next $n-2$ rows from second derivative continuity.
- Two rows from zero second derivative at first/last data point.

# Learning Outcomes – Integration

- Develop algorithms to apply concepts of numerical integration (e.g. Newton-Cotes and Gaussian quadrature methods) to discrete data.
- Know the mathematics associated with the trapezium rule. Gaussian quadrature.

# Numerical vs Analytical Integration

Not all integrals can be solved analytically, but can be either approximated or precisely calculated using numerical methods.

Consider a generalised integral:

$$I = \int_a^b g(x)\ dx$$

The value of the integral, $I$, corresponds to the area under the graph of the *integrand*, $g(x)$, between $x = a$ and $x = b$.

# Numerical Integration

Numerical integration divides the integration range into a number of subintervals and sums the area of these individual subintervals.

- Known integrand: the subintervals can be chosen.
- Unknown integrand: the subintervals coincide with the sampled data. The integrand is approximated by an interpolating function, $f(x) \approx g(x)$.

# Newton-Cotes Methods

Newton-Cotes methods fit a polynomial to each subinterval, $p_i(x)$, and evaluates the geometric area under each polynomial.

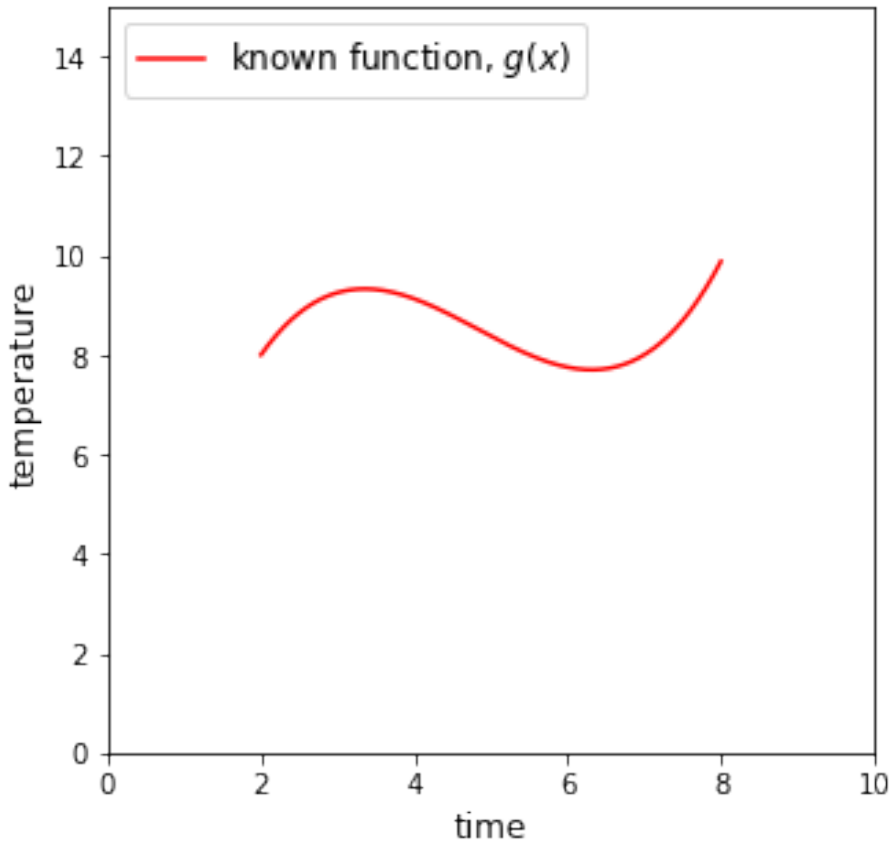
There are two commonly used variants:

- Trapezium method: order 1 polynomial
- Simpson's method: order 2 polynomial

# Numerical Integration - Example

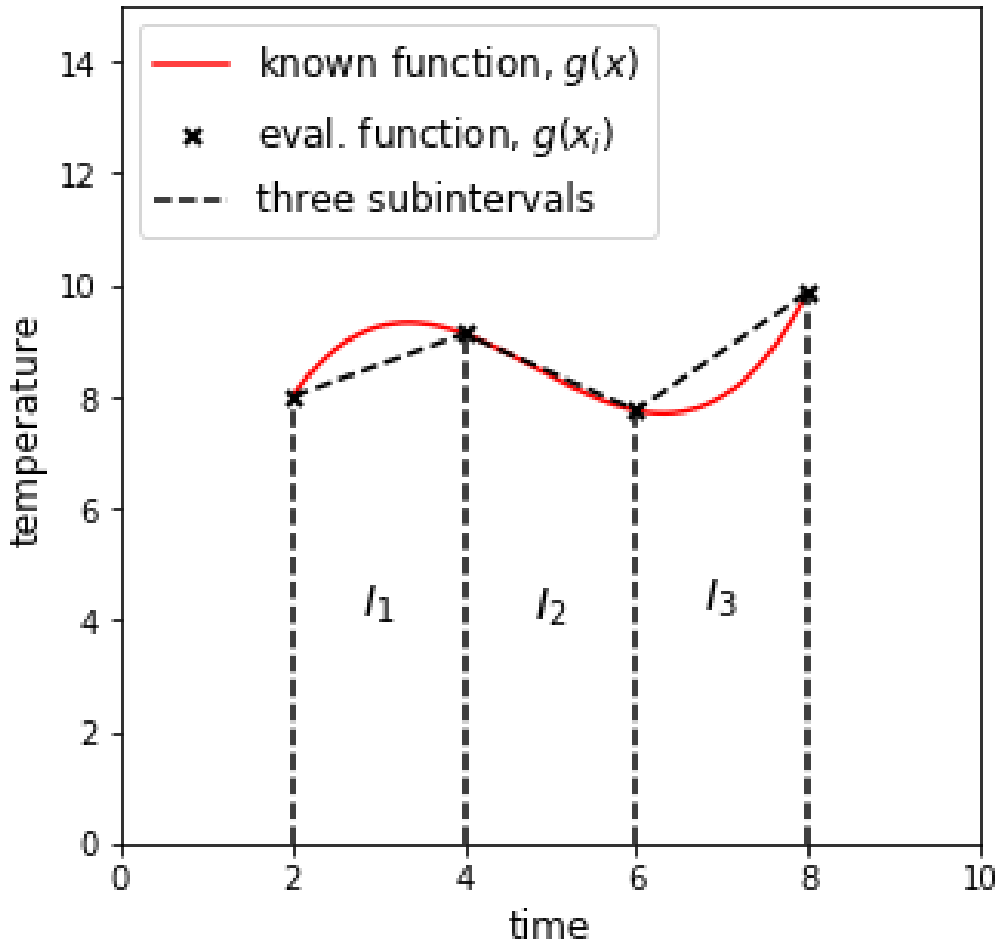
Consider numerical integration of a known integrand:

$$I = \int_2^8 g(x)\ dx$$



# Numerical Integration - Example

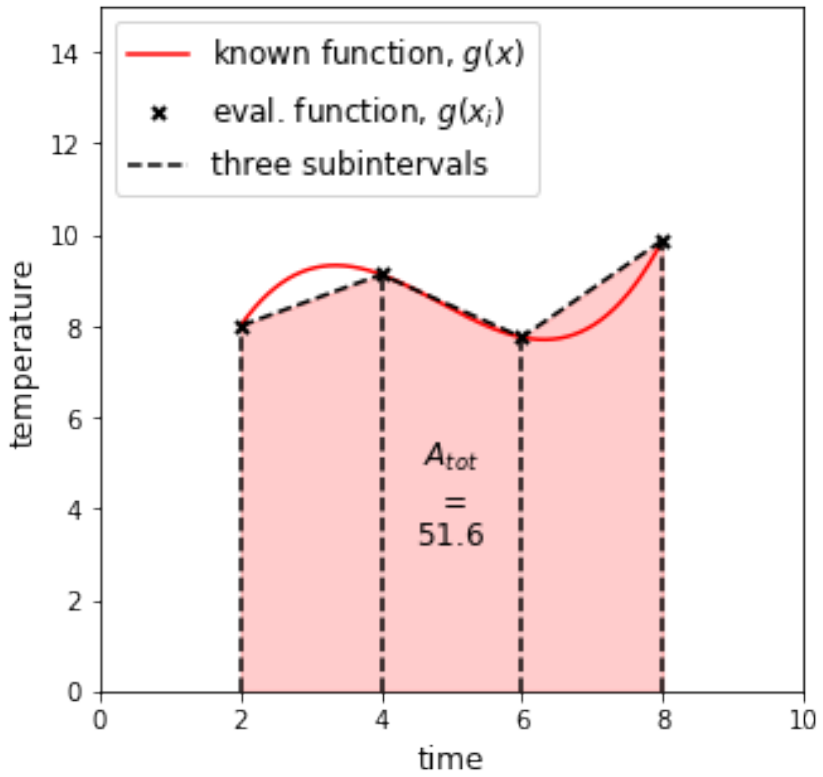
Numerical approximation of the integral using subintervals:



# Numerical Integration - Example

The subinterval area sum, $A_{tot}$, approximates the integral, $I$:

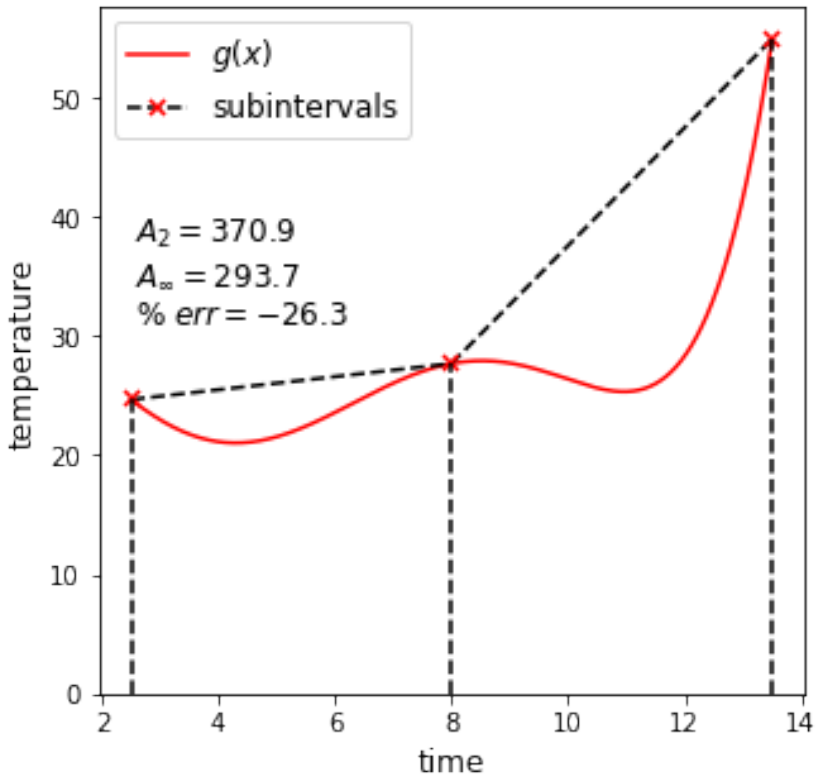
$$I \approx A_{tot} = \sum_{i=0}^{n-2} I_i$$

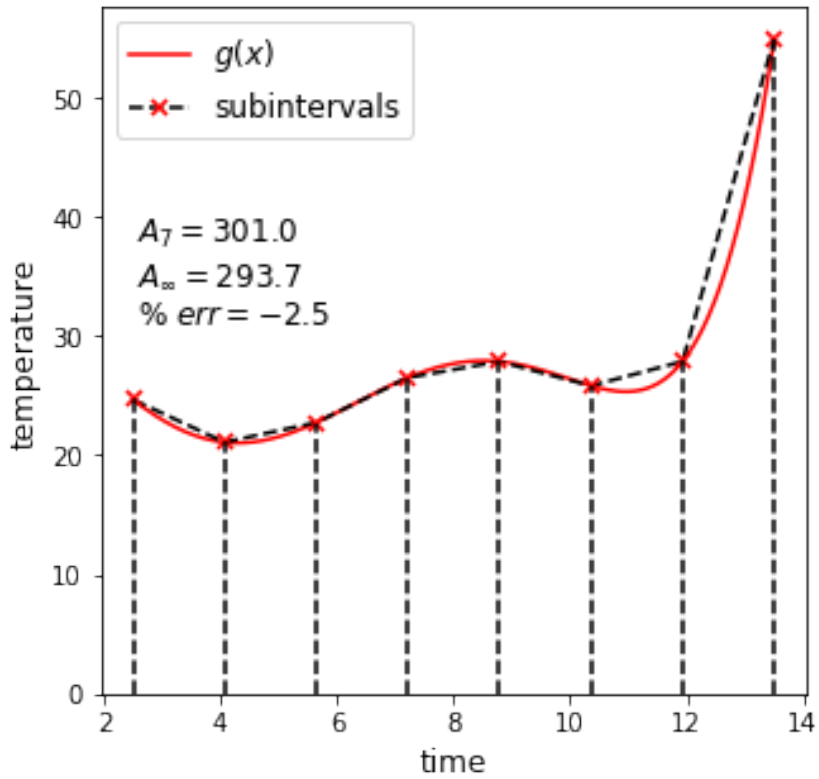


# Numerical Accuracy of Newton-Cotes

There is typically a trade-off between numerical accuracy and computational expense for Newton-Cotes methods.

For example:



# Trapezium Method

Fit an order 1 polynomial to each subinterval i.e. the same as in piecewise linear interpolation. The area for subinterval $i$ is:

$$I_i = \frac{y(x_i) + y(x_{i+1})}{2}(x_{i+1} - x_i)$$

# Simpson's Method

Fit an order 2 polynomial to each subinterval. This polynomial will exactly match the data at either end of the subinterval, as well as an mid-point.

We generally won't go into much detail on this method.

# Simpson's Method

Fit an order 2 polynomial, $p_i(x)$, to subinterval $i$ that exactly matches the data points at either end of the subinterval, as well as an imaginary mid-point:

$$f_i(x_i) = a{x_i}^2 + bx_i + c$$
$$f_i\left(\frac{x_i + x_{i+1}}{2}\right) = a{x_i}^2 + bx_i + c$$
$$f_i(x_{i+1}) = a{x_{i+1}}^2 + bx_{i+1} + c$$

i.e. the same as in piecewise linear interpolation. The area for subinterval $i$ is:

$$I_i = \frac{y(x_i) + y(x_{i+1})}{2}(x_{i+1} - x_i)$$