ENGSCI 233 – Computational Techniques and Computer Systems

Week 3: Sampled Data (Interpolation) Integration

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

THE UNIVERSITY OF AUCKLAND NEW ZEALAND | ENGINEERING

Department of Engineering Science

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

Sections

1. Introduction and Learning Outcomes

2. Interpolation - Theory

3. Interpolation - Polynomial Fitting

4. Polynomial Fitting - Worked Example

5. Interpolation - Piecewise Linear

6. Interpolation - Cubic Splines

7. Integration Overview

8. Newton-Cotes

Introduction

Sampled Data

numpy
matplotlib
plotly

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

Interpolation is a supervised machine learning problem.

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. $i = 0, \ldots, n-1$
- 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. $y(x_i) = y_i$

$$\mathcal{D} = \{(x_i, y_i)\}_{i=0}^{n-1}$$ set notation of data set $\mathcal{D}$

features

labels / targets

$x_i \in \mathbb{R}$

$y_i \in \mathbb{R}$ regression

Example Data

i	x_i	y_i
0	x_0	y_0
1	x_1	y_1
$\vdots$	$\vdots$	
$n-1$	x_{n-1}	y_{n-1}

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
i	0	1	2	3	4	5	6	7

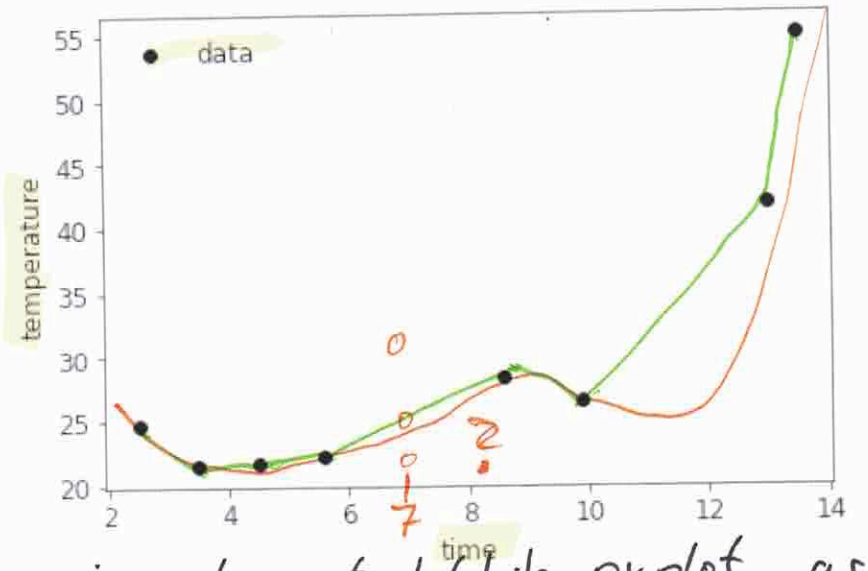
- The independent variable is time, $t = x$
- The dependent variable is temperature, $T = y$
- The set of $n = 8$ measurement points are $t_i = x_i$
- The set of $n = 8$ measurement values are $T_i = T(t_i)$.

$y_i = y(x_i)$

Data Visualisation

```
import numpy as np
x = np.array([2.5, 3.5, 4.5, ..... 13.5])
y = np.array([24.7, 21.5, ..... 54.8])
```

A basic visualisation of the example sampled data:




```
import matplotlib.pyplot as plt
plt.plot(x, y, 'o', label = 'data')
```

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


```
plt.xlabel('time')
plt.ylabel('temperature')
plt.legend()
```

Introduction to Interpolation

Interpolation $\hat{=}$ „function approximation"

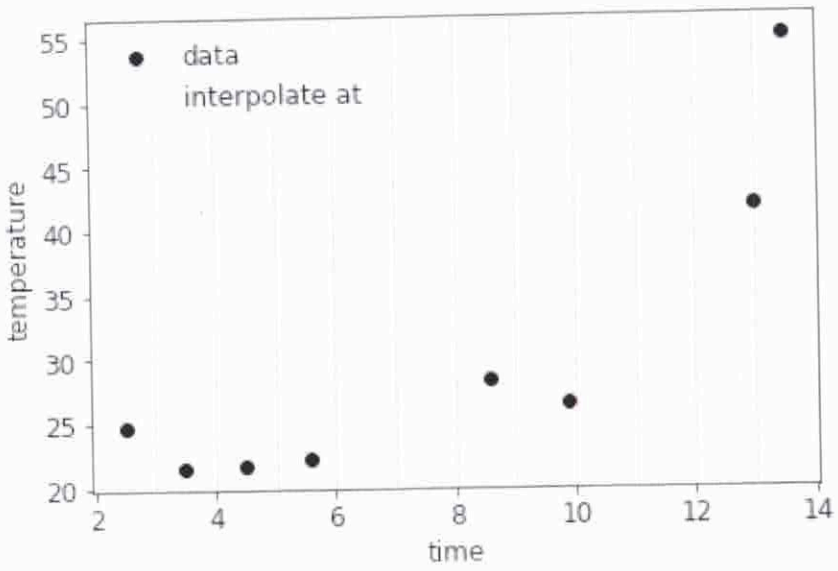
Often we require an estimate of the dependent variable at a point where no data are available. For example, what is $T(7)$?

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

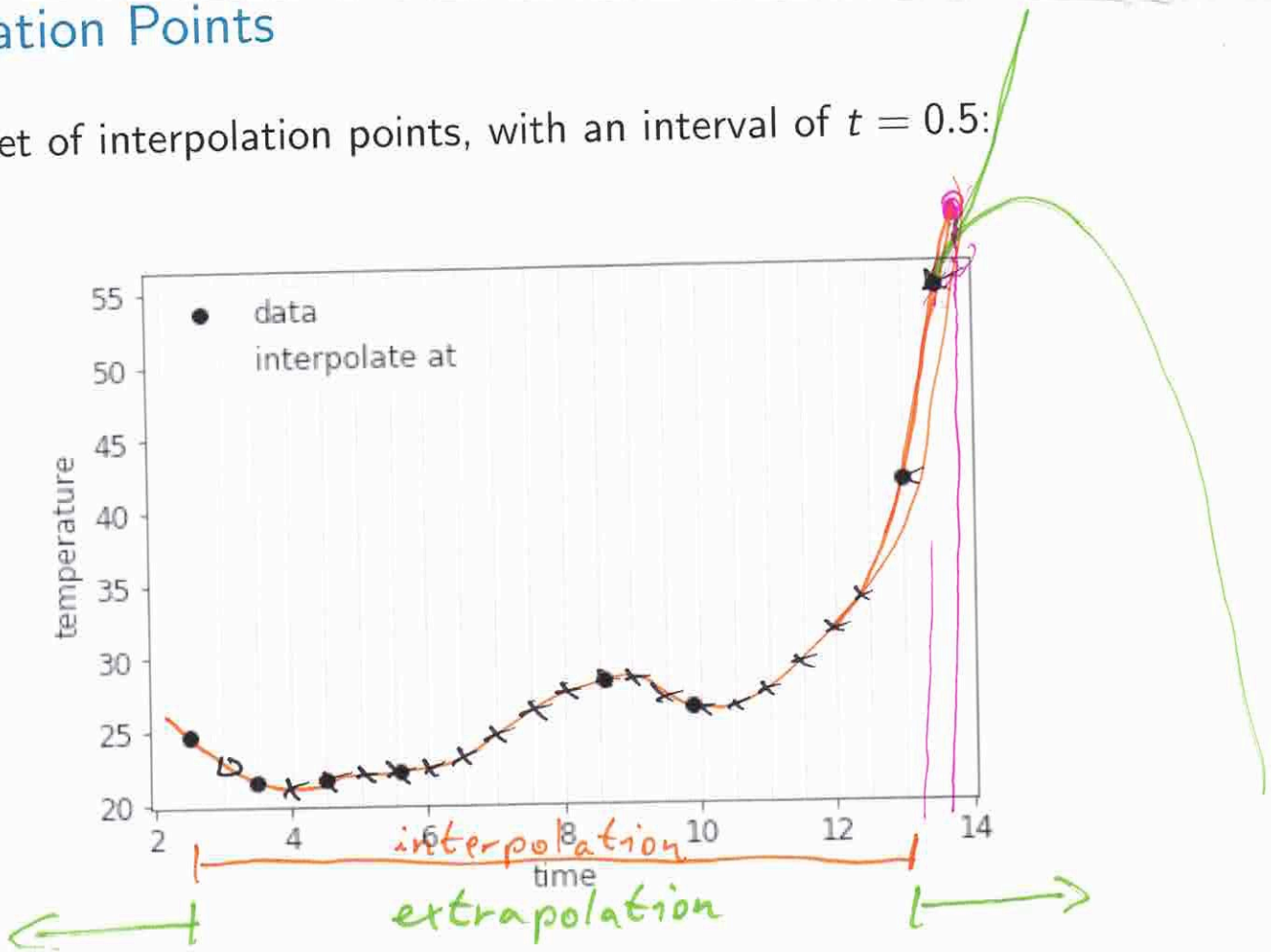
One possible set of interpolation points, with an interval of $t = 1$:



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, $f(x_j) = y(x_j)$.

↳ function approximation given (x_0, y_0), (x_1, y_1), ⋮ (x_{n-1}, y_{n-1})

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

polynomial regression (ENGSCI 211)
quadratic regression $m = 2$

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

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

$$f(x) = a_0 x^0 + a_1 x^1 + a_2 x^2 + \ldots a_m x^m$$

Or in *summation notation*:

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

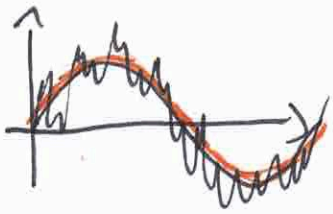
$x^0 = 1$
$x^1 = x$

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':

$\hat{Y}_i = F(x_i)$

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

residual r_i

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

where we find the polynomial coefficients that minimise RSS.

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

mean absolute error

$$MAE = \frac{1}{n} \sum_{i=0}^{n-1} |Y(x_i) - F(x_i)|$$

mean squared error

$$MSE = \frac{1}{n} RSS$$

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, $A\vec{a} = \vec{b}$:

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

- A is known as the Vandermonde matrix.
- $\vec{a}$ 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 $A\vec{\alpha} = \vec{b}$ using a linear algebra solver.

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