L25: Interpolation

Join the dots

Lucas A. J. Bastien

E7 Spring 2017, University of California at Berkeley

March 20, 2017

Version: release

Announcements

Lab 09 is due on March 24 at 12 pm (noon)

Today:

- Interpolation (chapter 14)
  - Introduction and motivation
  - Nearest neighbor interpolation
  - Linear interpolation
  - Lagrange polynomial
  - Cubic splines

Wednesday:

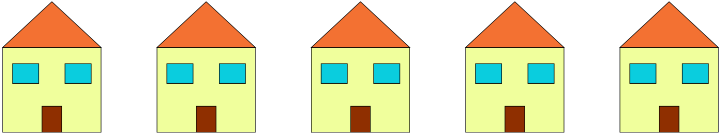
- Series (chapter 15)
- Presentation of the final programming project

Friday:

- Discussion

Introduction to interpolation (well water contamination)

Consider the following (equally-spaced) houses and the corresponding water quality measurements (mock values, arbitrary units)



8 10 ?? 40

What is the water quality value in the fourth house from the left?

(A) 50

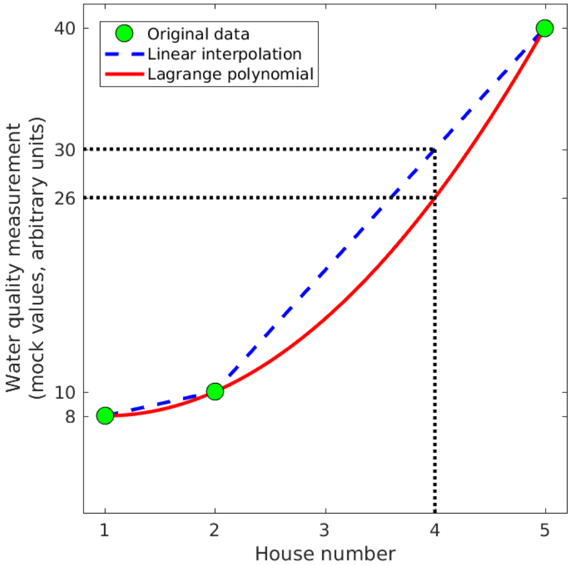
(B) 30

(C) 26

(D) 40

(E) I am not sure

Introduction to interpolation (well water contamination)

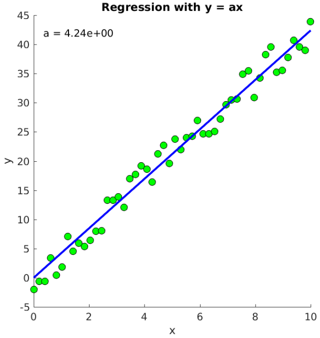


Introduction to interpolation

Interpolation: starting from discrete data, estimate values between data points, by using functions that go through all the data points

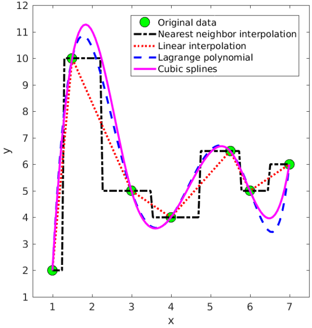
Linear regression

- Draw best-fit line going through a cloud of points
- Fitted line does **not always** go through all the data points



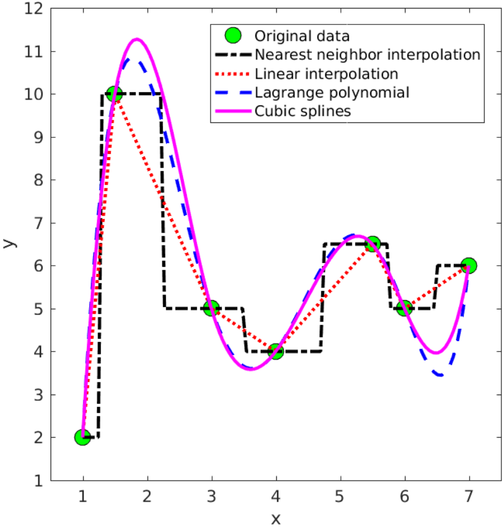
Interpolation

- "Join the dots", filling missing values between data points
- Interpolation line **does** go through all the data points



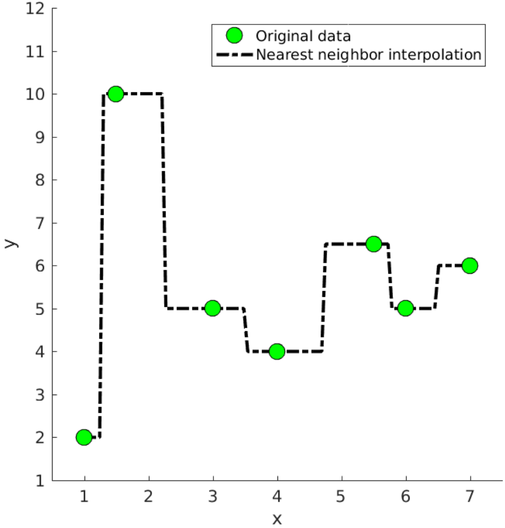
Introduction to interpolation

Today, we learn about **four interpolation methods**: **nearest neighbor**, **linear interpolation**, **Lagrange polynomial**, and **cubic splines**



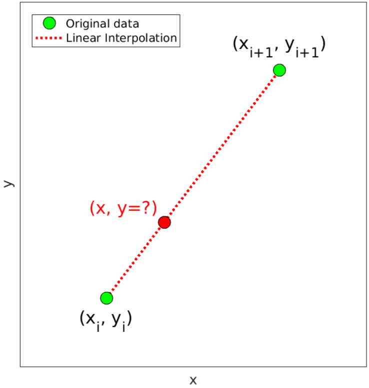
Nearest neighbor interpolation:

Associate, to each value of x, the y_i value associated with the closest x_i



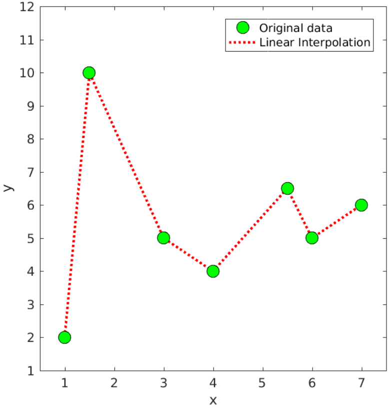
Linear interpolation:

Link any two consecutive points with a straight line



$$y = y_i + \frac{y_{i+1} - y_i}{x_{i+1} - x_i}(x - x_i) \quad \text{for} \;\; x \in [x_i, x_{i+1}]$$

Linear interpolation:
Link any two consecutive points with a straight line

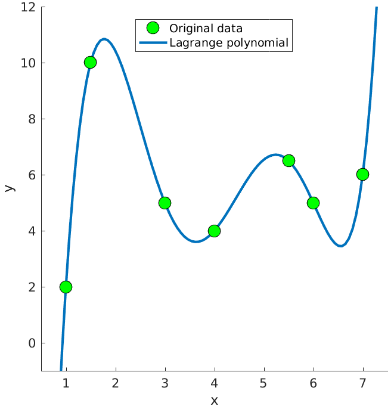


The equation varies between different pairs of points

Lagrange polynomial

Consider a set of m data points $(x_i, y_i), i = \{1, 2, \ldots, m\}$, such that all the x_i's are different from each other

Lagrange polynomial:
Polynomial of least degree that goes through all the points



Lagrange polynomial

Consider a set of m data points $(x_i, y_i), i = \{1, 2, \ldots, m\}$, such that all the x_i's are different from each other. How many different polynomials of degree m that go through all these data points can we find?

(A) Zero

(B) One and only one

(C) An infinite number

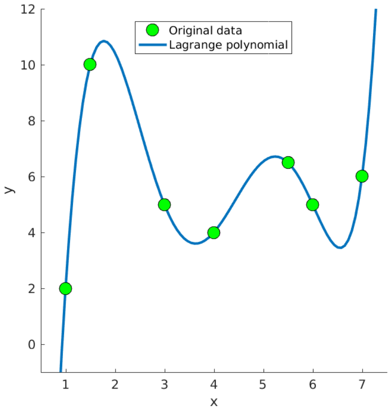
(D) It depends on the data

Lagrange polynomial

Consider a set of m data points $(x_i, y_i), i = \{1, 2, \ldots, m\}$, such that all the x_i's are different from each other. We calculate the Lagrange polynomial for this data set. Which of the following statements are true?

- **(A)** $\boxed{\text{The degree of the polynomial is } m-1 \text{ or less}}$
- **(B)** The degree of the polynomial is m
- **(C)** The degree of the polynomial is m or more
- **(D)** The degree of the polynomial is $m+1$ or more
- **(E)** None of the above

Lagrange polynomial L:
Polynomial of least degree that goes through all the points



$$L(x) = \sum_{i=1}^{m} y_i l_i(x)$$

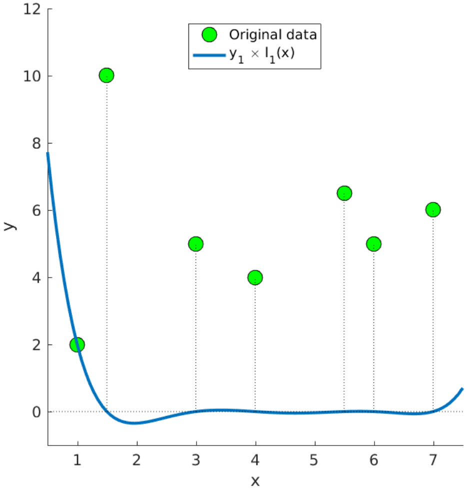
l_i: basis polynomial, such that:

$$l_i(x_j) = \begin{cases} 1 & \text{if } j = i \\ 0 & \text{if } j \neq i \end{cases}$$

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

Lagrange polynomial

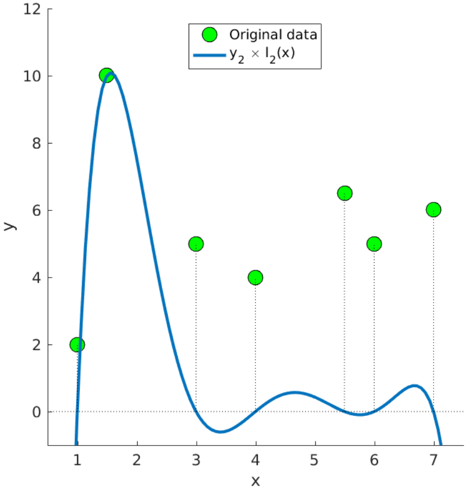
l_1 is zero at all the x_i's except for x_1, where it is equal to 1
$y_1 \times l_1$ is zero at all the x_i's except for x_1, where it is equal to y_1



$$y_1 l_1(x) = y_1 \prod_{\substack{j=1 \\ j \neq 1}}^{m} \frac{x - x_j}{x_1 - x_j}$$

Lagrange polynomial

l_2 is zero at all the x_i's except for x_2, where it is equal to 1
$y_2 \times l_2$ is zero at all the x_i's except for x_2, where it is equal to y_2

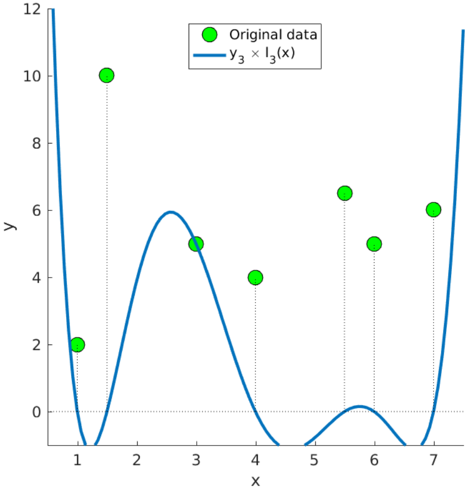


$$y_2 l_2(x) = y_2 \prod_{\substack{j=1 \\ j \neq 2}}^{m} \frac{x - x_j}{x_2 - x_j}$$

Lagrange polynomial

l_3 is zero at all the x_i's except for x_3, where it is equal to 1

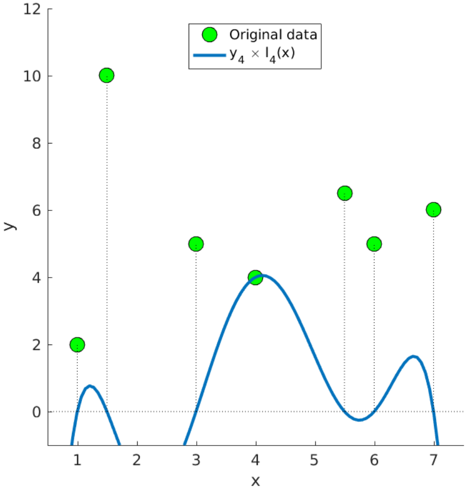
$y_3 \times l_3$ is zero at all the x_i's except for x_3, where it is equal to y_3



$$y_3 l_3(x) = y_3 \prod_{\substack{j=1 \\ j \neq 3}}^{m} \frac{x - x_j}{x_3 - x_j}$$

Lagrange polynomial

l_4 is zero at all the x_i's except for x_4, where it is equal to 1
$y_4 \times l_4$ is zero at all the x_i's except for x_4, where it is equal to y_4

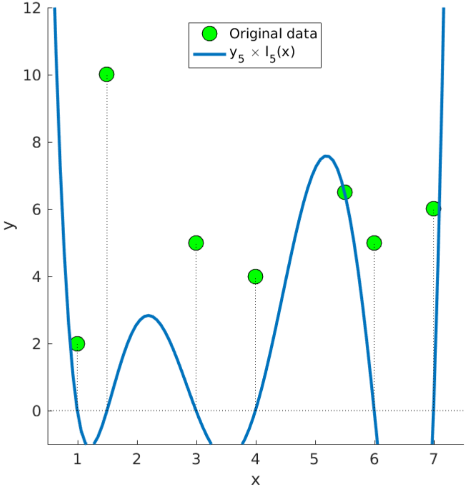


$$y_4 l_4(x) = y_4 \prod_{\substack{j=1 \\ j \neq 4}}^{m} \frac{x - x_j}{x_4 - x_j}$$

Lagrange polynomial

l_5 is zero at all the x_i's except for x_5, where it is equal to 1

$y_5 \times l_5$ is zero at all the x_i's except for x_5, where it is equal to y_5

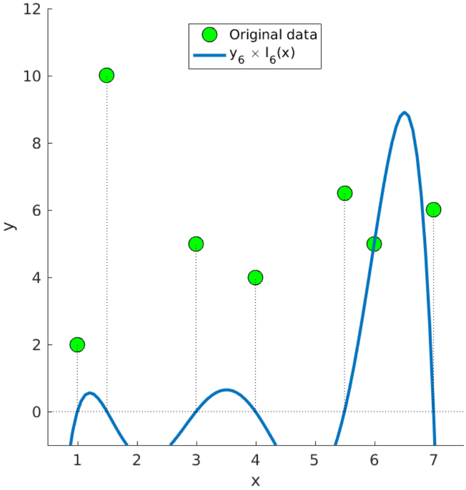


$$y_5 l_5(x) = y_5 \prod_{\substack{j=1 \\ j \neq 5}}^{m} \frac{x - x_j}{x_5 - x_j}$$

Lagrange polynomial

l_6 is zero at all the x_i's except for x_6, where it is equal to 1

$y_6 \times l_6$ is zero at all the x_i's except for x_6, where it is equal to y_6

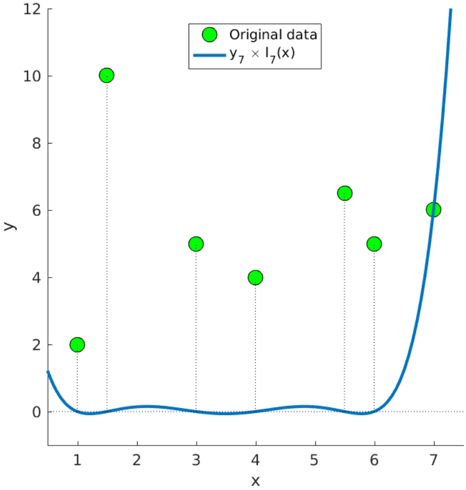


$$y_6 l_6(x) = y_6 \prod_{\substack{j=1 \\ j \neq 6}}^{m} \frac{x - x_j}{x_6 - x_j}$$

Lagrange polynomial

l_7 is zero at all the x_i's except for x_7, where it is equal to 1

$y_7 \times l_7$ is zero at all the x_i's except for x_7, where it is equal to y_7



$$y_7 l_7(x) = y_7 \prod_{\substack{j=1 \\ j \neq 7}}^{m} \frac{x - x_j}{x_7 - x_j}$$

Lagrange polynomial

Alternative approach: let Matlab do the work

For m data points, we are looking for a polynomial of degree $m-1$ or smaller:

$$L(x) = a_{m-1}x^{m-1} + \cdots + a_1 x + a_0$$

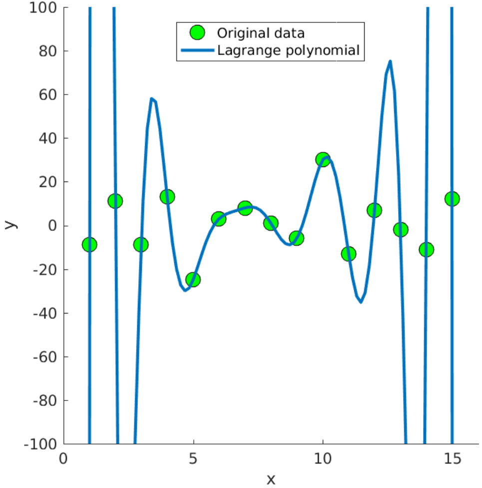
that goes through all the data points *i.e.*

$$\begin{aligned}
&a_{m-1}x_1^{m-1} + \cdots + a_1 x_1 + a_0 = y_1 \\
&a_{m-1}x_2^{m-1} + \cdots + a_1 x_2 + a_0 = y_2 \\
&\ldots \\
&a_{m-1}x_m^{m-1} + \cdots + a_1 x_m + a_0 = y_m
\end{aligned}$$

We have a system of m linear algebraic equations with m unknowns (the coefficients $a_0, a_1, \ldots, a_{m-1}$). **We know how to solve such a system using Matlab!**

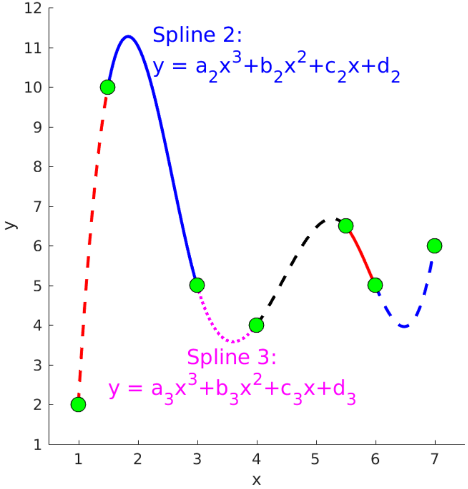
Lagrange polynomials: not good with many data points

Lagrange polynomials should generally not be used on a data set that has many points, as the polynomial will likely feature **many “wiggles”, sometimes of high magnitude**. For example:



Cubic splines: introduction

General idea: fit cubic polynomials ("splines"), one separate spline between each pair of consecutive points, and make the different splines connect smoothly



Cubic splines: introduction

General idea: fit cubic polynomials ("splines"), one separate spline between each pair of consecutive points, and make the different splines connect smoothly

Consider a set of m data points $(x_i, y_i), i = \{1, 2, \ldots, m\}$, such that all the x_i's are in order and different from each other

Group of 2 points	Equation of the spline	Unknowns
$(x_1, y_1), (x_2, y_2)$	$a_1x^3 + b_1x^2 + c_1x + d_1$	a_1, b_1, c_1, d_1
$(x_2, y_2), (x_3, y_3)$	$a_2x^3 + b_2x^2 + c_2x + d_2$	a_2, b_2, c_2, d_2
...	...	...

Number of splines: $m - 1$
Number of unknown coefficients: $4(m - 1)$

We need to write $4(m - 1)$ independent equations

Cubic splines: first set of equations

Obtain the first set of equations by **making each spline go through its respective two data points**:

First spline:

$$a_1 x_1^3 + b_1 x_1^2 + c_1 x_1 + d_1 = y_1$$
$$a_1 x_2^3 + b_1 x_2^2 + c_1 x_2 + d_1 = y_2$$

Second spline:

$$a_2 x_2^3 + b_2 x_2^2 + c_2 x_2 + d_2 = y_2$$
$$a_2 x_3^3 + b_2 x_3^2 + c_2 x_3 + d_2 = y_3$$

And so on . . .

We can write $2(m - 1)$ such equations

Cubic splines: second set of equations

Obtain the second set of equations by **making the splines' first derivatives match at the connection points**:

Connection between the first and second splines ($x = x_2$):

$$3a_1x_2^2 + 2b_1x_2 + c_1 = 3a_2x_2^2 + 2b_2x_2 + c_2$$

Connection between the second and third splines ($x = x_3$):

$$3a_2x_3^2 + 2b_2x_3 + c_2 = 3a_3x_3^2 + 2b_3x_3 + c_3$$

And so on . . .

We can write $(m - 2)$ such equations
We have written $2(m - 1) + (m - 2)$ equations so far

Cubic splines: third set of equations

Obtain the third set of equations by **making the splines' second derivatives match at the connection points**:

Connection between the first and second splines ($x = x_2$):

$6a_1x_2 + 2b_1 = 6a_2x_2 + 2b_2$

Connection between the second and third splines ($x = x_3$):

$6a_2x_3 + 2b_2 = 6a_3x_3 + 2b_3$

And so on . . .

We can write $(m - 2)$ such equations
We have written $2(m - 1) + 2(m - 2)$ equations so far

Cubic splines: last set of equations, and solving the system

We need 2 more equations. These two equations depend on the application. Often, one sets the second derivative of the corresponding splines to be zero at $x = x_1$ (first point) and $x = x_m$ (last point):

$$6a_1x_1 + 2b_1 = 0$$
$$6a_{m-1}x_m + 2b_{m-1} = 0$$

We have written $2(m-1) + 2(m-2) + 2 = 4(m-1)$ equations
We have $4(m-1)$ unknown coefficients to calculate

Write the system of algebraic linear equations in matrix form, set up the corresponding matrices in Matlab, and "let Matlab do the rest"

Lagrange polynomial and cubic splines are different

Cubic splines are generally preferable over the Lagrange polynomial when the data set has many points (less "wiggly"). For example: