Math 311

Numerical Methods

3.1: Interpolation and the Lagrange Polynomial

Fitting points to a curve

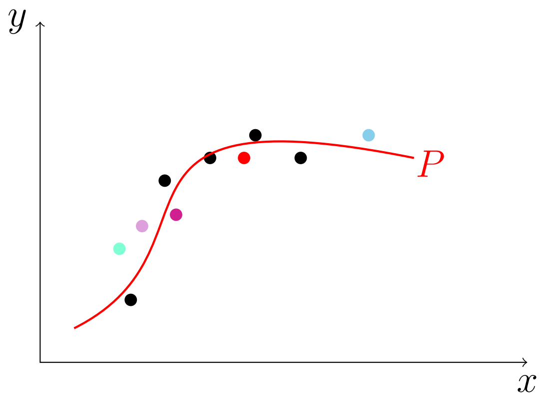
S. K. Hyde

Burden and Faires, any ed.

Winter 2024

1 Introduction

- Suppose you have several points on a graph:



- How do you find the polynomial that passes through points?
- Many methods. We will learn how to do it.
- We will start with a linear model: $P(x) = ax + b$
- You've learned this method in algebra class.
- You plug in both points and solve for the coefficients a and b.

1.1 Example (Fitting a line between two points)

- Let's find the line that passes through (x_0, y_0) and (x_1, y_1). So:

$$y_0 = P(x_0) = ax_0 + b \qquad\qquad y_1 = P(x_1) = ax_1 + b$$

- So set $b = b$ which leads to $\quad y_0 - ax_0 = y_1 - ax_1$
- followed by $\quad ax_1 - ax_0 = y_1 - y_0$
- Solving for a yields: $\quad a = \left(\dfrac{y_1 - y_0}{x_1 - x_0}\right)$
- We can then find b from: $b = y_1 - ax_1$.
- Which gives $\quad b = y_1 - \left(\dfrac{y_1 - y_0}{x_1 - x_0}\right)x_1$
- This finally gives $P(x) = \left(\dfrac{y_1 - y_0}{x_1 - x_0}\right)x + y_1 - \left(\dfrac{y_1 - y_0}{x_1 - x_0}\right)x_1$
- What do you think? Is it easy?
- Yes, not too bad, but what about adding more points?
- This is difficult to extend to more than two points.
- We can make it easier if we approach it from a different perspective.

2 Lagrange Polynomials

- Let's start with an easier form for $P(x)$:

$$P(x) = \left(\frac{x - x_1}{x_0 - x_1}\right) y_0 + \left(\frac{x - x_0}{x_1 - x_0}\right) y_1$$

- This particular form interpolates the two points. You can easily see that when you plug in x_0 you get

$$P(x_0) = \cancelto{1}{\left(\frac{x_0 - x_1}{x_0 - x_1}\right)} y_0 + \cancelto{0}{\left(\frac{x_0 - x_0}{x_1 - x_0}\right)} y_1 = y_0$$

- and when you plug in x_1 you get

$$P(x_1) = \cancelto{0}{\left(\frac{x_1 - x_1}{x_0 - x_1}\right)} y_0 + \cancelto{1}{\left(\frac{x_1 - x_0}{x_1 - x_0}\right)} y_1 = y_1$$

- The form is what makes it easy to do. You don't have to solve for any of the coefficients because they solve themselves!

- Plus, it extends to more variables with ease!

2.1 Example (Lagrange polynomial with n=1) (two points) (a line)

- Find the line that connects $(1, 5)$ and $(2, 7)$. Let $(x_0, y_0) = (1, 5)$ and $(x_1, y_1) = (2, 7)$.
- It follows that the polynomial (a line in this case) is:

$$\begin{aligned} P(x) &= \left(\frac{x \;\; - x_1}{x_0 - x_1}\right)(y_0) + \left(\frac{x \;\; - x_1}{x_0 - x_1}\right)(y_1) \\ &= \left(\frac{x - 2}{1 - 2}\right)(5) + \left(\frac{x - 2}{1 - 2}\right)(7) \\ &= -5(x - 2) + 7(x - 1) && \text{(Easiest form to write)} \\ &= 2x + 3 && \text{(simplified form (not needed))} \end{aligned}$$

- This automatically solves for the polynomial that exactly fits the points.
- To generalize, we add more points to the equation. For example, if we have three points, then we will have the sum of three terms, for 7 points, seven terms.
- Each one of these terms will cancel out all the terms for the other points and keep its own term. Then it repeats for all the other points.
- As an example, let's explain the 2 point case above:

- We want to write the polynomial like this:

$$P(x) = L_{2,0}(x)y_0 + L_{2,1}(x)y_1$$

- In this case, $L_{2,0}(x) = \left(\dfrac{x \; - x_1}{x_0 - x_1}\right)$ and $L_{2,1} = \left(\dfrac{x \; - x_0}{x_1 - x_0}\right)$.
- Note that

$$L_{2,0}(x_0) = 1 \text{ and } L_{2,0}(x_1) = 0 \text{ and}$$
$$L_{2,1}(x_0) = 0 \text{ and } L_{2,1}(x_1) = 1.$$

- It “picks” the right x at the right time!
- Let’s extend it to three points. This time, we’d like to write

$$P(x) = L_{2,0}(x)y_0 + L_{2,1}(x)y_1 + L_{2,2}(x)y_2$$

  where the L functions pick the right x’s at the right time.
- So, how do we do that for three points?

2.2 Lagrange Polynomial with $n = 2$ (three points)

$$P(x) = L_{3,0}(x)y_0 + L_{3,1}(x)y_1 + L_{3,2}(x)y_2$$

- Here's our goal:
  - We want to make $P(x_0) = y_0$, which means we need $\boxed{\begin{array}{l} L_{2,0}(x_0) = 1 \\ L_{2,1}(x_0) = 0 \\ L_{2,2}(x_0) = 0 \end{array}}$
  - We want to make $P(x_1) = y_1$, which means we need $\boxed{\begin{array}{l} L_{2,0}(x_1) = 0 \\ L_{2,1}(x_1) = 1 \\ L_{2,2}(x_1) = 0 \end{array}}$
  - We want to make $P(x_2) = y_2$, which means we need $\boxed{\begin{array}{l} L_{2,0}(x_2) = 0 \\ L_{2,1}(x_2) = 0 \\ L_{2,2}(x_2) = 1 \end{array}}$
- It follows that

$$L_{2,0}(x) = \frac{(x - x_1)(x - x_2)}{(x_0 - x_1)(x_0 - x_2)} \qquad L_{2,1}(x) = \frac{(x - x_0)(x - x_2)}{(x_1 - x_0)(x_1 - x_2)} \qquad L_{2,2}(x) = \frac{(x - x_0)(x - x_1)}{(x_2 - x_0)(x_2 - x_1)}$$

- In general, suppose we have $(n+1)$ distinct points

$$(x_0, y_0), (x_1, y_1), \cdots, (x_n, y_n)$$

- The points may also come from a function. In that case, $f(x_k) = y_k = P(x_k)$.

Lagrange Polynomial

$$P(x) = L_{n,0}(x)f(x_0) + L_{n,1}(x)f(x_1) + \cdots + L_{n,k}(x)f(x_k) + \cdots + L_{n,n}(x)f(x_n)$$

$$\text{or simply } P(x) = \sum_{k=0}^{n} L_{n,k}(x)f(x_k), \text{ where } L_{n,k}(x) \text{ is defined below}$$

Definition as $L_{n,k}$

$$L_{n,k}(x) = \frac{(x - x_0)(x - x_1)\cdots(x - x_{k-1})(x - x_{k+1})\cdots(x - x_n)}{(x_k - x_0)(x_k - x_1)\cdots(x_k - x_{k-1})(x_k - x_{k+1})\cdots(x_k - x_n)}$$

$$= \prod_{\substack{j=0 \\ j \neq k}}^{n} \left(\frac{x - x_k}{x_j - x_k} \right)$$

- This makes it so that $L_{n,k}(x_j) = \begin{cases} 0, & \text{if } j \neq k \\ 1, & \text{if } j = k \end{cases}$
- Notice that if one of the terms is missing in it.

$$L_{n,k}(x) = \frac{(\ x\ - x_0)(\ x\ - x_1)\cdots(\ x\ - x_{k-1})(\ x\ - x_k)(\ x\ - x_{k+1})\cdots(\ x\ - x_n)}{(x_k - x_0)(x_k - x_1)\cdots(x_k - x_{k-1})\underbrace{(x_k - x_k)}_{\text{This term is removed!}}(x_k - x_{k+1})\cdots(x_k - x_n)}$$

- If you were to plug in x_k into the removed term, it would cause a division by zero.
- That's why the "$j \neq k$" is included in this form (take out that problem!)

$$L_{n,k}(x) = \prod_{\substack{j=0 \\ j\neq k}}^{n} \left(\frac{x\ - x_k}{x_j - x_k} \right)$$

- We usually leave out the n in $L_{n,k}(x)$ and write $L_k(x)$. How good is this function?

Theorem 3.3 - Validity of Lagrange Interpolating Polynomial

Theorem. *If $x_0, x_1, \cdots, x_n$ are distinct numbers on $[a, b]$ and $f \in C^{n+1}[a, b]$, then for each x in $[a, b]$, a number $\xi(x)$ in (a, b) exists with*

$$f(x) = P(x) + \frac{f^{(n+1)}(\xi(x))}{(n+1)!}(x - x_0)(x - x_1) \cdots (x - x_n),$$

where P is the interpolating polynomial defined on the previous slide.

- The theorem states that $P(x)$ fits the function as good as possible!
- The error formula in Theorem 3.3 is an important theoretical result because Lagrange polynomials are used extensively for deriving numerical differentiation and integration methods (Chapter 4).
- Compare the error term in Theorem 3.3 to Taylor's Theorem error term:

$$R_n(x) = \frac{f^{(n+1)}(\xi(x))}{(n+1)!}(x - x_0)(x - x_1) \cdots (x - x_n),$$

$$R_{Taylors}(x) = \frac{f^{(n+1)}(\xi(x))}{(n+1)!}(x - x_0)^{n+1}$$

- They are similar, but the Taylor polynomial concentrates all the known information at x_0.
- Whereas, the Lagrange polynomial spreads out the error information among the $n+1$ different points.
- Next example will be how to fit a polynomial to 4 points.

2.3 Example (Four points)

- Let's use 4 points from $f(x) = \frac{1}{x}$. It follows that $L_k(x)$ are:

x	$f(x)$
$2/3$	$3/2$
1	1
3	$1/3$
4	$1/4$

$$L_0(x) = \frac{(x-1)(x-3)(x-4)}{(2/3-1)(2/3-3)(2/3-4)} = -\frac{27}{70}(x-1)(x-3)(x-4)$$

$$L_1(x) = \frac{(x-2/3)(x-3)(x-4)}{(1-2/3)(1-3)(1-4)} = \frac{1}{6}(3x-2)(x-4)(x-3)$$

$$L_2(x) = \frac{(x-1)(x-2/3)(x-4)}{(3-1)(3-2/3)(3-4)} = -\frac{1}{14}(3x-2)(x-4)(x-1)$$

$$L_3(x) = \frac{(x-1)(x-2/3)(x-3)}{(4-1)(4-2/3)(4-3)} = \frac{1}{30}(3x-2)(x-1)(x-3)$$

- Note the colored numbers match vertically. It follows a pattern
- So it follows that $P(x) = 3/2 \cdot L_0(x) + 1 \cdot L_1(x) + 1/3 \cdot L_2(x) + 1/4 \cdot L_3(x)$
- If you simplify to standard form, then $P(x) = -\frac{1}{8}x^3 + \frac{13}{12}x^2 - \frac{73}{24}x + \frac{37}{12}$
- This polynomial perfectly fits the table. Try it on $P(2/3)$, $P(1)$, $P(3)$, or $P(4)$.
- Note that $f(2) = 1/2$, whereas $P(2) = -\frac{1}{8}(2)^3 + \frac{13}{12}(2)^2 - \frac{73}{24}(2) + \frac{37}{12} = \frac{1}{3}$
- Here's a graph of it: `https://www.desmos.com/calculator/p3brjfbvdu`
- What is it like to expand this to FIVE points?

2.4 Example (Five points)

- Let's use the extra point $(2, 1/2)$. It follows that $L_k(x)$ are:

x	$f(x)$
$2/3$	$3/2$
1	1
3	$1/3$
4	$1/4$
2	$1/2$

$$L_0(x) = \frac{(x-1)(x-3)(x-4)(x-2)}{(2/3-1)(2/3-3)(2/3-4)(2/3-2)} = \frac{81}{280}(x-1)(x-3)(x-4)(x-2)$$

$$L_1(x) = \frac{(x-2/3)(x-3)(x-4)(x-2)}{(1-2/3)(1-3)(1-4)(1-2)} = -\frac{1}{6}(3x-2)(x-3)(x-4)(x-2)$$

$$L_2(x) = \frac{(x-2/3)(x-1)(x-4)(x-2)}{(3-2/3)(3-1)(3-4)(3-2)} = -\frac{1}{14}(3x-2)(x-1)(x-4)(x-2)$$

$$L_3(x) = \frac{(x-2/3)(x-1)(x-3)(x-2)}{(4-2/3)(4-1)(4-3)(4-2)} = \frac{1}{60}(3x-2)(x-1)(x-3)(x-2)$$

$$L_4(x) = \frac{(x-2/3)(x-1)(x-3)(x-4)}{(2-2/3)(2-1)(2-3)(2-4)} = \frac{1}{8}(3x-2)(x-1)(x-3)(x-4)$$

- It follows that

$$P(x) = 3/2 \cdot L_0(x) + 1 \cdot L_1(x) + 1/3 \cdot L_2(x) + 1/4 \cdot L_3(x) + 1/2 \cdot L_4(x)$$

- If you simplify to standard form, then $P(x) = \frac{3}{8}x^4 - \frac{2}{3}x^3 + \frac{125}{48}x^2 - \frac{55}{12}x + \frac{43}{12}$
- This polynomial perfectly fits the table.
- Note that this time $f(2) = 1/2$!
- Here's a graph of it: `https://www.desmos.com/calculator/yf28jveamw`
- Let's go over to it and explore what it looks like.

- Note that adding another point required us to START OVER. Knowledge of the current polynomial cannot be used to help with the next one.
- Which is the best polynomial? Remember that

$$f(x) = P(x) + \frac{f^{(n+1)}(\xi(x))}{(n+1)!}(x - x_0)(x - x_1)\cdots(x - x_n),$$

  That means that the error is bound by:

$$|f(x) - P(x)| \leqslant \frac{\max\limits_{x\in[a,b]} |f^{(n+1)}(x)|}{(n+1)!}|(x - x_0)(x - x_1)\cdots(x - x_n)|$$

- This bound works well IF we know $f^{(n+1)}(x)$. What if you don't know them?
- When we do not know the derivatives, then there is NO way to tell which polynomial is the best.
- All we can do is use the rule that higher degree polynomial gives the smallest error.
- Example 3 in Sec 3.1 (5th ed,) gives an example, where a lower degree polynomial worked better. But without knowledge of the derivatives, we would not know that.
- There is a fix for adding points!! Or at least a partial fix
- We can generate better approximations recursively (this is called Neville's Method). (Neville Longbottom????)

Standard for naming polynomials for a set of points

Definition.

- *Let f be defined at $x_0, x_1, \cdots, x_n$, and suppose that $m_1, m_2, \cdots, m_k$ are k distinct integers with $0 \leqslant m_i \leqslant n$ for each i.*
- *The Lagrange polynomial that agrees with f at the k points $x_{m_1}, x_{m_2}, \cdots, x_{m_k}$ is denoted $P_{m_1, m_2, \cdots, m_k}$.*

- Examples: (each are the Lagrange polynomial that fits the indicated points.)
  - $P_{0,1,5,6}(x)$ agrees at the points x_0, x_1, x_5, and x_6.
  - $P_{8,9}(x)$ agrees at the points x_8 and x_9.
- How do we use this?
- First, a theorem that shows how to combine two previous polys to generate a new one.

Theorem 3.5

Theorem. *Let f be defined at $x_0, x_1, \cdots, x_k$ and let x_j and x_i be two distinct numbers in this set. (so $0 \leqslant i, j \leqslant k$) Then*

$$P(x) = \frac{(x - x_j)P_{0,1,\cdots,j-1,j+1,\cdots,k}(x) - (x - x_i)P_{0,1,\cdots,i-1,i+1,\cdots,k}(x)}{x_i - x_j}$$

describes the kth Lagrange polynomial that interpolates f at the $k+1$ points $x_0, x_1, \cdots, x_k$

- More detail:
  - $P_{0,1,\cdots,j-1,j+1,\cdots,k}(x)$ agrees with every point BUT x_j.
  - $P_{0,1,\cdots,i-1,i+1,\cdots,k}(x)$ agrees with every point BUT x_i.
- Suppose $i = 2$, $j = 3$, and $k = 6$. Then it follows that
  - $P_{0,1,2,4,5,6}(x)$ doesn't agree with x_3.
  - $P_{0,1,3,4,5,6}(x)$ doesn't agree with x_2. We can combine them to:
  - $P_{0,1,2,3,4,5,6}(x) = \dfrac{(x - x_2)P_{0,1,2,4,5,6}(x) - (x - x_3)P_{0,1,3,4,5,6}(x)}{x_3 - x_2}$

- This is very versatile, and can be used to add a point anywhere in the set. However, we will focus on adding points at the end. (e.g. go from 0,1,2 to 0,1,2,3).
- In this case, the m_i's follow in succession. (no missing gaps) (e.g. 2,3,4,5 or 0,1,2, etc.)
- Then we can create an algorithm that generates successive approximations from previous approximations.

Neville's Method

Theorem.

- *Let* $0 \leqslant i \leqslant j$ *denote the interpolating polynomial of degree* j *on the* $j+1$ *numbers* $x_{i-j}, x_{i-j-1}, \cdots, x_{i-1}, x_i$.
- *In other words,* $Q_{i,j} = P_{i-j,i-j+1,\cdots,i-1,i}$
- *Then it follows that*

$$Q_{i,j}(x) = \frac{(x - x_{i-j})Q_{i,j-1} - (x - x_i)Q_{i-1,j-1}}{x_i - x_{i-j}}$$

Matrix Describing Neville's Algorithm

Using the notation for Neville's method given above, we have the following matrix

x	y	First Order	Second Order	Third Order	Fourth Order
x_0	$y_0 = P_0$				
x_1	$y_1 = P_1$	$P_{0,1}$			
x_2	$y_2 = P_2$	$P_{1,2}$	$P_{0,1,2}$		
x_3	$y_3 = P_3$	$P_{2,3}$	$P_{1,2,3}$	$P_{0,1,2.3}$	
x_4	$y_4 = P_4$	$P_{3,4}$	$P_{2,3,4}$	$P_{1,2,3,4}$	$P_{0,1,2,3,4}$

It is easier to program if we change it to: $Q_{i,j} = P_{i-j,i-j+1,\ldots,i-1,i}$.

x	y	First Order	Second Order	Third Order	Fourth Order
x_0	$Q_{0,0}$				
x_1	$Q_{1,0}$	$Q_{1,1}$			
x_2	$Q_{2,0}$	$Q_{2,1}$	$Q_{2,2}$		
x_3	$Q_{3,0}$	$Q_{3,1}$	$Q_{3,2}$	$Q_{3,3}$	
x_4	$Q_{4,0}$	$Q_{4,1}$	$Q_{4,2}$	$Q_{4,3}$	$Q_{4,4}$

Notes on Neville's Algorithm and R

Definition.

- *Unfortunately, R does not allow zero indices in its programming.*
- *Using Python or C would probably be better.*
- *However, we can fix it.*
- *We need to adjust the algorithm by shifting away from zero.*
- *Adjustments: start the counters at 2 and increase the vector entry* **x[i-j]** *with* **x[i-j+1]**
- *Also populate the first column before performing the loop below.*

```
for (i in 2:(n+1)) {
    for (j in 2:i) {
```

$$Q[i,j] = \frac{(xs - x[i-j+1])Q[i,j-1] - (xs - x[i])Q[i-1,j-1]}{x[i] - x[i-j+1]}$$

```
    }
}
```

2.5 Example: Neville's Method

- We will estimate the value of the function at $x = 2$ using Neville's Method.
- The following data set are the values of the Digamma function at 4 points.
- The real value at $x = 2$ is 0.422784335098467.
- How good will Neville's work?
- Let's start by just using nodes 1 & 2.

Node	x	$f(x)$
0	0.5	−1.9635100260214231
1	1.5	0.0364899739785769
2	2.5	0.7031566406452434
3	3.5	1.1031566406452433

R Code

```
> x=seq(.5,by=1,length=4)
> y=digamma(x)
> A=cbind(x,y)
> neville(A,2,1:2) # Only use 2 points now.
$table
                 f(x)       first order
1.5 0.0364899739785769               NA
2.5 0.7031566406452434 0.36982330731191

$interp
[1] 0.36982330731191
```

- The estimate is 0.3698 with error of 0.052961027
- Now let's use the full capability of it.

- Now we will show using all the nodes.
- The true value $x = 2$ is 0.422784335098467.
- This shows first, second, and third order approximations
- Don't specify any nodes this time.

Node	x	$f(x)$
0	0.5	−1.963510026021423l
1	1.5	0.0364899739785769
2	2.5	0.7031566406452434
3	3.5	1.1031566406452433
4	4.5	1.3888709263595289

R Code

```
$table
                    f(x)         first order        second order
0.5 -1.9635100260214231                   NA                  NA
1.5  0.0364899739785769 1.0364899739785769                  NA
2.5  0.7031566406452434 0.3698233073119102 0.5364899739785769
3.5  1.1031566406452433 0.5031566406452435 0.4031566406452435
4.5  1.3888709263595289 0.6745852120738149 0.4602994977881005
           third order       fourth order
0.5                 NA                 NA
1.5                 NA                 NA
2.5                 NA                 NA
3.5 0.4698233073119102                 NA
4.5 0.4126804501690530 0.4483947358833388

$interp
[1] 0.4483947358833388
```

- Every single number is a different approximation to the function evaluated at 2.
- The best value is the bottom right of the matrix. (a fourth order approx)

2.6 Inverse Interpolation

- Inverse Interpolation is an alternative to using Bisection or Newton's.
- Inverse Quadratic Interpolation is used in Mueller's and Brent's method.
- We can find a zero of the function by evaluating the inverse at 0 (zero $= f^{-1}(x)$).
- To estimate the inverse, we just switch x and y in the matrix.

R Code

```
> neville(A,0)
$table
                     f(x)         first order       second order        third order
-1.96351002602142     0.5                   NA                 NA                 NA
0.0364899739785769    1.5   1.48175501301071 18                NA                 NA
0.703156640645243     2.5   1.44526503903213 47 1.454886851852138                 NA
1.10315664064524      3.5   0.742108398386891 6 1.469319571082143 1.464127761279594
1.38887092635953      4.5  -0.361048242258353 4 1.873325778135062 1.458418666306317
                          fourth order
-1.96351002602142                   NA
0.0364899739785769                  NA
0.703156640645243                   NA
1.10315664064524                    NA
1.38887092635953     1.460783909438539

$interp
[1] 1.460783909438539
```

- The zero is 1.460783909438539.