

Math 104A - Numerical Analysis I

APPROXIMATION OF FUNCTIONS

JEAN-HYUN PARK

UNIVERSITY OF CALIFORNIA SANTA BARBARA

FALL 2022



Introduction

Polynomial Interpolation

Before we begin

- Solution to Math HW1: GauchoSpace later today.
- Some answers to “How could secant method be superior to Newton?”
 - ▶ This exercise does not make much sense and I would not be able to explain this to someone.
 - ▶ Secant method requires less iterations to locate the zero.
 - ▶ It makes sense b/c Newton is “one-sided,” while the secant restricts the domain.
 - ▶ **I don't think the exercise makes sense**, because each one is useful in different contexts with different given information.
 - ▶ I don't know how to say one is better than the other, but secant method needs two initial guesses but Newton needs one.
 - ▶ I don't think this is a bad question as it tests for understanding of both, but superiority in what sense should definitely be specified.
 - ▶ I think secant is better but Newton's method is easier to understand.
 - ▶ It makes sense b/c secant method takes less time when computing using computer
 - ▶ Secant method is more flexible. It requires only 1 evaluation per iteration whereas Newton's method requires 2. compared to Newton's method

PROBLEM OF INTEREST

Throughout the section, we want to answer the problem (and some important properties): given the data below, find a polynomial $y = p(x)$ of minimal degree *interpolating* it.

x	x_0	x_1	x_2	\cdots	x_n
y	y_0	y_1	y_2	\cdots	y_n

- **Subjective questions:**
Does this problem make sense? What should we check to make this problem meaningful? What do your guts tell you before even start studying it?

POLYNOMIAL INTERPOLATION

Theorem (Unique interpolation theorem)

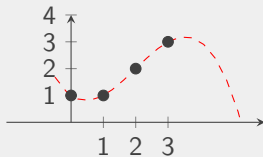
If x_0, x_1, \dots, x_n are distinct real, for arbitrary values y_0, y_1, \dots, y_n , there is a unique polynomial $p \in \Pi_n$ such that $p(x_i) = y_i$ ($0 \leq i \leq n$).

Proof 1.

Vandermonde – next few slides .



- Notation: $\Pi_n := \{ \text{polynomials of degree at most } n \}$.
- Notice that the degrees of freedom match: $(n+1)$ values to interpolate and $(n+1)$ coefficients we can tune in $p(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0$.



VANDERMONDE MATRIX

Idea: Brute force.

Set $p(x) = a_0 + a_1x + a_2x^2 + \cdots + a_nx^n$, and require the conditions.

$$p(x_0) = a_0 + a_1x_0 + a_2x_0^2 + \cdots + a_nx_0^n = y_0$$

$$p(x_1) = a_0 + a_1x_1 + a_2x_1^2 + \cdots + a_nx_1^n = y_1$$

$$\vdots \quad \quad \quad \vdots \quad \quad \quad \vdots \quad \quad \quad \vdots$$

$$p(x_n) = a_0 + a_1x_n + a_2x_n^2 + \cdots + a_nx_n^n = y_n$$

In matrix form,

$$\begin{bmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^n \\ 1 & x_1 & x_1^2 & \cdots & x_1^n \\ 1 & x_2 & x_2^2 & \cdots & x_2^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^n \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

- The coefficient matrix is called **Vandermonde matrix**.

VANDERMONDE MATRIX

Theorem

$\det(V) = \prod_{0 \leq i < j \leq n} (x_j - x_i)$, where V is the Vandermonde matrix:

$$V = \begin{bmatrix} 1 & x_0 & x_0^2 & \dots & x_0^n \\ 1 & x_1 & x_1^2 & \dots & x_1^n \\ 1 & x_2 & x_2^2 & \dots & x_2^n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \dots & x_n^n \end{bmatrix}.$$

Proof.

1. $\det(V)$ is a polynomial in x_0, x_1, \dots, x_n .
2. If $x_0 = x_1$, the first two rows are identical. Thus, $\det(V) = 0$. The factor theorem asserts $(x_0 - x_1)$ divides $\det(V)$. (Think of x_0 as a variable, say x , and x_1 as a number, say 1, and plug in $x = 1$.) Do the same for $x_i = x_j$ ($i \neq j$) and conclude $(x_i - x_j)$ divides $\det(V)$.
Therefore, $\det(V) = (\text{something}) \prod_{0 \leq i < j \leq n} (x_j - x_i)$.

VANDERMONDE MATRIX

Proof.

3. Recall Leibniz formula for the determinant: sum of \pm (product of entries taken from distinct columns while scanning rows from the top to the bottom). '+' is assigned when the order of column index chosen is an *even permutation* of $(0, 1, \dots, n)$ and '-' when it is an *odd permutation*.
4. Observe that 'something' must be a constant since the order of the polynomial is $n(n+1)/2 = 0 + 1 + 2 + \dots + n$ from both Leibniz formula and the product form.
5. Comparing the term $x_1 x_2^2 \dots x_n^n$, we realize that the constant must be 1: this term appears only once with '+' in the Leibniz formula and we have (something) $x_1 x_2^2 \dots x_n^n$ by expanding (something) $\prod_{0 \leq i < j \leq n} (x_j - x_i)$ choosing only x_j 's.



POLYNOMIAL INTERPOLATION

Theorem (Unique interpolation theorem)

If x_0, x_1, \dots, x_n are distinct real, for arbitrary values y_0, y_1, \dots, y_n , there is a unique polynomial $p \in \Pi_n$ such that $p(x_i) = y_i$ ($0 \leq i \leq n$).

Proof 1.

Since the nodes are distinct, the determinant of the Vandermonde matrix $\det(V) = \prod_{0 \leq i < j \leq n} (x_j - x_i)$ nonzero, hence the matrix is invertible. Therefore, we have a unique solution $[a_0, a_1, \dots, a_n]^T$ in the Vandermonde system for any prescribed $[y_0, y_1, \dots, y_n]^T$. That is, $p(x) = a_0 + a_1x + a_2x^2 + \dots + a_nx^n \in \Pi_n$ is the unique polynomial we want. □

NEWTON FORM INTERPOLATION

Theorem (Unique interpolation theorem - duplicate)

If x_0, x_1, \dots, x_n are distinct real, for arbitrary values y_0, y_1, \dots, y_n , there is a unique polynomial $p \in \Pi_n$ such that $p(x_i) = y_i$ ($0 \leq i \leq n$).

Proof 2.

Board work.



Example

Following the previous proof, find a polynomial (of minimal degree) interpolating

x	0	1	2	3
y	1	1	2	3

- The way the polynomial organized in the proof is called **Newton form**.

HORNER'S ALGORITHM: EVALUATING POLYNOMIALS

By storing coefficients, we can only *encode* a polynomial

$$\begin{aligned} p(x) &= 1 + 0 \cdot x + \frac{1}{2}x(x-1) - \frac{1}{6}x(x-1)(x-2) \\ &= -\frac{1}{6}x^3 + x^2 - \frac{5}{6}x + 1. \end{aligned}$$

We need to compute the output just to know one function value.

For practical reasons, the following **nested multiplication** or **Horner's algorithm** is better than following the math expression.

$$\left(\left(-\frac{1}{6}(x-2) + \frac{1}{2} \right) (x-1) \right) x + 1$$

In algorithm form, this reads much nicer:

Algorithm 1: Horner's algorithm for polynomial evaluations

```
u ← ck;  
for i ← k − 1 to 0 do  
    | u ← (t − xi)u + ck;
```

- This is **only for evaluating** a polynomial after finding an interpolation. Don't mix this with how to find Newton form interpolations.
- Multiplications are more expensive than additions in computing. Count the multiplications to see the difference.
- This is purely computational. Mathematically, they are the same.

LAGRANGE INTERPOLATION

Idea: Find a basis of Π_n that makes interpolating procedure simple.
In particular, if we can find $\ell_i(x) \in \Pi_n$ such that

$$\ell_i(x_j) = \delta_{ij}, \quad (1)$$

then, (we will call the way it's written **Lagrange form**)

$$p(x) = \sum_{i=0}^n y_i \ell_i(x).$$

Definition (Lagrange basis or cardinal functions)

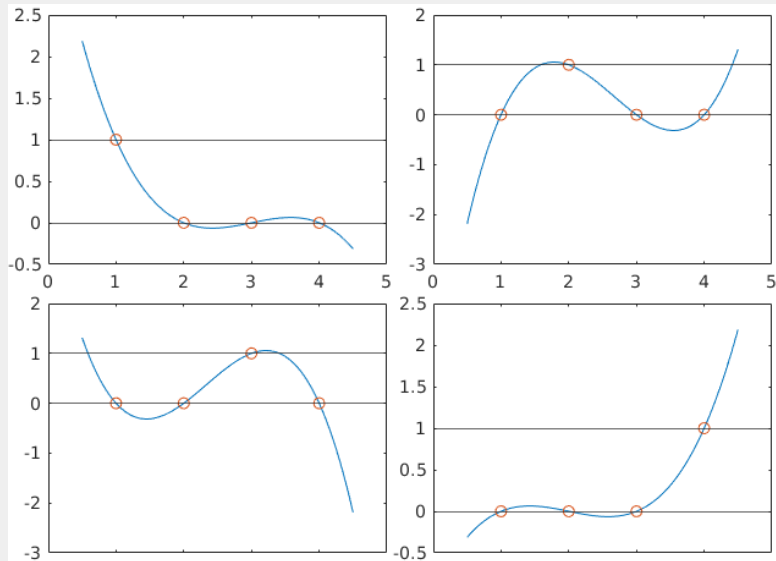
For a given set of distinct abscissas $\{x_i\}_{i=0}^n$,

$$\ell_i = \prod_{j \neq i} \frac{x - x_j}{x_i - x_j}, \quad (0 \leq i \leq n) \quad (2)$$

are called **Lagrange basis** or **cardinal functions** associated to $\{x_i\}_{i=0}^n$.

- Π_n is a vector space.
For review, think of the following: recall the definition, useful concepts to better know such a space (find those for Π_n).
- **Notation: (Kronecker delta)**
$$\delta_{ij} = \begin{cases} 1 & (i = j) \\ 0 & (i \neq j) \end{cases}$$
- 'abscissas' means horizontal coordinates $\{x_i\}_{i=0}^n$.

LAGRANGE INTERPOLATION



- ℓ_0 (top left), ℓ_1 (top right), ℓ_2 (bottom left), ℓ_3 (bottom right)
- **Notation:** ℓ_i 's will be reserved to be Lagrange basis functions from now on.

LAGRANGE INTERPOLATION

Theorem

If a set of functions $\{f_i(x)\}_{i=0}^n$ satisfies $f_i(x_j) = \delta_{ij}$, then it is linearly independent.

Proof.

Board work. □

Corollary

Lagrange basis is indeed a basis.

Proof.

Since $\dim(\Pi_n) = \#\{\ell_i(x)\}_{i=0}^n = n + 1$, it suffices to show linear independence. Observe $\ell_i(x_j) = \prod_{k \neq i} \frac{x_j - x_k}{x_i - x_k} = 0$ if $j \neq i$ (one of the numerator is zero) and, if $j = i$, $\ell_i(x_i) = \prod_{k \neq i} \frac{x_i - x_k}{x_i - x_k} = 1$. Linear independence follows from the previous theorem.

- $p(x) = \sum_{i=0}^n y_i \ell_i(x)$ means that just putting the data as coordinates (or coefficients) of the Lagrange basis, you have the interpolation.
- This result can be considered 3rd proof of the polynomial interpolation theorem.

LAGRANGE INTERPOLATION

Example

Following the previous proof, find a polynomial (of minimal degree) interpolating

x	0	1	2	3
y	1	1	2	3

POLYNOMIAL INTERPOLATION COMPARISON

	Vandermonde	Lagrange	Newton
Basis	$1, x, x^2 \dots$	$\ell_i = \prod_{j \neq i} \frac{x - x_j}{x_i - x_j}$	$1, (x - x_0), (x - x_0)(x - x_1), \dots$
Theory	Some algebraic beauty	Convenient for Lagrange interpolation (i.e., only function values involved)	Effective for Hermite interpolation (i.e., also derivatives involved)
Numerical	Inaccurate and inefficient: the matrix is <i>ill-conditioned</i> and inverting a matrix is among expensive computations	Efficient when nodes are fixed but possibly the data to fit changes (Lagrange basis depends only on the nodes)	Efficient when nodes gets added (a newly added term does not affect the previous interpolations). Also, finding coefficients can be efficient when equipped with divided difference . ¹
Evaluation algorithm	Horner	Some algorithms exist	Horner

¹Text in blue: next topics.

POLYNOMIAL INTERPOLATION ERROR

Theorem

Let $x_0, x_1, \dots, x_n \in [a, b]$ be distinct nodes, $f \in C^{n+1}[a, b]$, and $p \in \Pi_n$ interpolating f at the nodes. For each $x \in [a, b]$, there is $\xi_x \in (a, b)$ such that

$$f(x) - p(x) = \frac{1}{(n+1)!} f^{(n+1)}(\xi_x) \prod_{i=0}^n (x - x_i)$$

Proof.

Board work.



- This theorem is of great importance later on when we analyze numerical methods. There will be no waste of time appreciating detailed aspects of this theorem.

Example

Find a bound on errors made by the polynomial interpolation of $f(x) = \sin(x)$ at 11 distinct nodes on $[0, 1]$. What if we require the nodes to be equally spaced?

- Warning: Equally spaced nodes are not always the best choice.

RUNGE'S PHENOMENON

Question: For a very smooth function, say, $f \in C^\infty[-1, 1]$, imagine what polynomial interpolations will be like if you use equally spaced nodes? What will happen as we increase the nodes?

Example (Runge's phenomenon)

Dynamic example of

$$\frac{1}{1 + 25x^2}$$

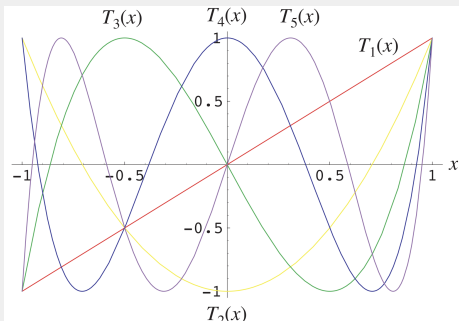
- Thank you, Cleve Moler, for this example.

Chebyshev Polynomials

Motivation: Though we will not be able to discuss the full picture, some “best” interpolation is related to **Chebyshev polynomials**.

Definition (Chebyshev polynomials - 1st kind)

$$T_0(x) = 1, \quad T_1(x) = x, \quad T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x) \quad (n \geq 1)$$



- When we talk about Chebyshev polynomials, we are interested in the domain $[-1, 1]$ though they are defined everywhere.

- First few are:

$$T_2(x) = 2x^2 - 1$$

$$T_3(x) = 4x^3 - 3x$$

$$T_4(x) = 8x^4 - 8x^2 + 1$$

- We omit “1st kind” from now on.

Chebyshev Polynomials

Theorem

For $x \in [-1, 1]$, we have

$$T_n(x) = \cos(n \cos^{-1} x) \quad (n \geq 0)$$

Proof.

Board work.



Corollary

$$|T_n(x)| \leq 1 \quad (-1 \leq x \leq 1)$$

$$T_n\left(\cos \frac{j\pi}{n}\right) = (-1)^j \quad (0 \leq j \leq n)$$

$$T_n\left(\cos \frac{2j-1}{2n}\pi\right) = 0 \quad (1 \leq j \leq n)$$



Chebyshev Polynomials

Theorem

Monic polynomials satisfy

$$\|p\|_{\infty} = \max_{-1 \leq x \leq 1} |p(x)| \geq 2^{1-n}.$$

Proof.

Board work. □

Theorem

If the nodes are zeros of T_{n+1} , then we have, for $|x| \leq 1$,

$$|f(x) - p(x)| \leq \frac{1}{2^n(n+1)!} \max_{|t| \leq 1} |f^{(n+1)}(t)|$$

Proof.

- Compare the new error formula with the previous one with a bad choice of nodes:

$$\begin{aligned} |f(x) - p(x)| &= \left| \frac{f^{(n+1)}(\xi_x)}{(n+1)!} \prod_{i=0}^n (x - x_i) \right| \\ &\leq \frac{\prod_{i=0}^n |x - x_i| |f^{(n+1)}(\xi_x)|}{(n+1)!} \\ &\leq \frac{2^{n+1} |f^{(n+1)}(\xi_x)|}{(n+1)!} \end{aligned}$$

RUNGE'S PHENOMENON REVISITED

Question: We saw a bad interpolating result for the Runge's function. What if we use Chebyshev nodes?

Example (Runge's phenomenon)

Dynamic example of

$$\frac{1}{1 + 25x^2}$$

- As the name suggests, **Chebyshev nodes** are the ones consist of the zeros of Chebyshev polynomials.

SUMMARY OF POLYNOMIAL INTERPOLATION

Here is some high level summary, which I believe is good enough for the very first course of numerical analysis.

- If a function is very well-behaving (like $\sin(x)$), reasonable polynomial interpolation (e.g., equally spaced ones) works well.
- Even if a function looks well-behaving (like $1/(1 + 25x^2)$), equally spaced nodes may not work. (To distinguish these two, we need to look through complex analysis lens.)
- If we choose a good set of nodes, the interpolation can be very satisfying (e.g., Runge's function with Chebyshev nodes)
- **(Weierstrass Approximation Theorem)** For any continuous function, we can find as good polynomial approximations as we please. (But it does not tell us how.) That is, let $f \in C[a, b]$, then, for $\forall \epsilon > 0$, there is a polynomial p such that $\|f - p\|_\infty < \epsilon$.

- As you have seen, polynomial interpolation is subtle and requires a deep dive for a better picture.
- Noticed $\frac{1}{1+25x^2}$ has singularities at $\pm \frac{\sqrt{-1}}{5}$. (We don't pursue this any further.)
- "Fix nodes first, then you can always find a bad function. Conversely, fix a function, then you can always find good nodes."

Polynomial Interpolation

Divided Differences

NEWTON FORM AND DIVIDED DIFFERENCES

Setting: given a function f and nodes x_0, x_1, \dots, x_n , find $p \in \Pi_n$ interpolating f , i.e., $p(x_i) = f(x_i)$, $(0 \leq i \leq n)$.

Example

Given nodes x_0, x_1, x_2 find the interpolating polynomial (of minimal degree) in Newton form. What does each coefficient depends on.

- Refresher: There is a unique interpolating polynomial given nodes and data. But the way it is written makes a huge (practical) difference.

NEWTON FORM AND DIVIDED DIFFERENCES

Definition (Divided differences)

Given a function f and nodes x_0, x_1, \dots, x_n , suppose $p(x) = \sum_{k=0}^n c_k q_k(x)$ is the polynomial interpolating f at the nodes in Newton form, where $q_k(x) = \prod_{j=0}^{k-1} (x - x_j)$, ($0 \leq k \leq n$) is the basis of Newton form. Then, **divided differences** are defined to be the coefficients

$$f[x_0, x_1, \dots, x_k] := c_k.$$

Corollary

Under the same assumptions as above,

$$\begin{aligned} p(x) &= \sum_{k=0}^n f[x_0, x_1, \dots, x_k] q_k(x) \\ &= \sum_{k=0}^n f[x_0, x_1, \dots, x_k] \prod_{j=0}^{k-1} (x - x_j). \end{aligned}$$

- **Convention:**
 $\sum_{k=0}^{-1} a_k = 0$ and $\prod_{k=0}^{-1} a_k = 1$. In words, “if a product or a sum does not make sense, assign it a value that has the same effect of doing nothing.”
- The notation $f[x_0, x_1, \dots, x_k]$ emphasizes it depend on f and the nodes only up to index k .

NEWTON FORM AND DIVIDED DIFFERENCES

Theorem (Recursive relation of divided differences)

$$f[x_0, x_1, \dots, x_n] = \frac{f[x_1, x_2, \dots, x_n] - f[x_0, x_1, \dots, x_{n-1}]}{x_n - x_0}$$

Proof.

Board work.



Example

Use the above formula to find a polynomial (of minimal degree) interpolating

x	0	1	2	3
y	1	1	2	3

- Mnemonic device: (a) looks similar to finite difference approximation of derivatives (in fact, this is exactly true for $f[x_0, x_1]$), (b) numerator index – last n minus first; numerator index – last one minus first one.

NEWTON FORM AND DIVIDED DIFFERENCES

Algorithm for divided differences is very efficient.

Algorithm 2: Divided differences

```
for i = 0 to n do
  |  $d_i \leftarrow f(x_i);$ 
end
for j = 0 to n do
  | for i = n to j do
  | |  $d_i \leftarrow (d_i - d_{i-1}) / (x_i - x_{i-j});$ 
  | end
end
```

Then, the interpolating polynomial is

$$p(x) = \sum_{i=0}^n d_i \prod_{j=0}^{i-1} (x - x_j).$$

- See the textbook pp. 331-332 for details of algorithm. We focus on other properties and applications of divided differences.

PROPERTIES OF DIVIDED DIFFERENCES

Theorem (Symmetry of divided differences)

If (z_0, z_1, \dots, z_n) is a permutation of (x_0, x_1, \dots, x_n) , then

$$f[z_0, z_1, \dots, z_n] = f[x_0, x_1, \dots, x_n]$$

Proof.

Board work.



Theorem (Error of polynomial interpolation)

Let $x_0, x_1, \dots, x_n \in [a, b]$ be distinct nodes and be $p \in \Pi_n$ interpolating f at the nodes. For each $t \in [a, b]$ different from the nodes, we have

$$f(x) - p(x) = f[x_0, x_1, \dots, x_n, t] \prod_{i=0}^n (x - x_i)$$

- Permutation means a shuffle.


PROPERTIES OF DIVIDED DIFFERENCES

Theorem (Error of polynomial interpolation)

Let $x_0, x_1, \dots, x_n \in [a, b]$ be distinct nodes and be $p \in \Pi_n$ interpolating f at the nodes. For each $t \in [a, b]$ different from the nodes, we have

$$f(t) - p(t) = f[x_0, x_1, \dots, x_n, t] \prod_{i=0}^n (t - x_i)$$

Proof.

Board work. 

- Permutation means a shuffle.

Theorem (Discrete derivatives)

Let $x_0, x_1, \dots, x_n \in [a, b]$ be distinct nodes. If $f \in C^n[a, b]$, there is $\xi \in (a, b)$ such that

$$f[x_0, x_1, \dots, x_n] = \frac{1}{n!} f^{(n)}(\xi)$$

Polynomial Interpolation

Hermite Interpolation

HERMITE INTERPOLATION

Motivation: We may want to interpolate not only the function values but also its slopes or curvatures too to get a high quality approximation. [Dynamic example](#)

Example

Find a polynomial that “interpolates” $f \in C^1$ satisfying $f(0) = 0, f(1) = 1, f'(\frac{1}{2}) = 2$.

- This example shows interpolating derivatives must be posed in a certain way if we want a simple, systematic solution.

HERMITE INTERPOLATION

Theorem

Given distinct nodes x_0, x_1, \dots, x_n , for any $c_{ij} \in \mathbb{R}$ ($\forall i, j$ that makes sense), there exists a unique polynomial $p \in \Pi_m$ satisfying

$$p^{(j)}(x_i) = c_{ij}, \quad (0 \leq j \leq k_i - 1, 0 \leq i \leq n),$$

where $k_i \geq 1$ and $m + 1 = k_0 + k_1 + \dots + k_n$.

Proof.

Board work. □

Example

Find a Hermite interpolation with only one node: $p^j(a) = c_j$,
($0 \leq j \leq n$)

- Notice that the degrees of freedom match:
 $\dim(\Pi_m) = \deg(p) + 1$
is equal to #conditions prescribed.
- The most useful special case is when $k_i = 2$
($\forall i$): there is a unique $p \in \Pi_{2n-1}$ such that
 $p(x_i) = y_i$ and
 $p'(x_i) = y'_i$.

HERMITE INTERPOLATION AND DIVIDED DIFFERENCES

Lemma

If $f \in C^1[a, b]$,

$$\lim_{x \rightarrow x_0} f[x_0, x] = f'(x_0)$$

Proof.

$$\lim_{x \rightarrow x_0} f[x_0, x] = \frac{f(x) - f(x_0)}{x - x_0} = f'(x_0)$$

■

□

This motivates the following definition.

Definition (Divided differences with repeated nodes)

If $f \in C^1[a, b]$, $f[x_0, x_0] := f'(x_0)$.

HERMITE INTERPOLATION AND DIVIDED DIFFERENCES

Similarly,

Lemma

If $f \in C^k[a, b]$ and $a \leq x_0 \leq x_1 \leq \cdots \leq x_k \leq b$,

$$\lim_{x_k \rightarrow x_0} f[x_0, x_1, \dots, x_k] = \frac{f^{(k)}(x_0)}{k!}$$

Proof.

Board work. □

Definition (Divided differences with repeated nodes)

For $k \geq 0$, if $f \in C^k[a, b]$,

$$f[\underbrace{x_0, \dots, x_0}_k] := \frac{f^{(k)}(x_0)}{k!}.$$

HERMITE INTERPOLATION AND DIVIDED DIFFERENCES

Example

Find the polynomial of minimal degree such that $p(x_0) = f(x_0)$, $p'(x_0) = f'(x_0)$, $p(x_1) = f(x_1)$, $p'(x_1) = f'(x_1)$.

Proof.

Boord work. ☐

Example

Find the polynomial of minimal degree such that $p(1) = 2$, $p'(1) = 3$, $p(2) = 6$, $p'(2) = 7$, $p''(2) = 8$.

Proof.

Boord work. ☐

- Be careful when working with second or higher degree: $f^{(k)}(x_i)/k!$ must be fed instead of $f^{(k)}(x_i)$!

ERROR OF HERMITE INTERPOLATION

Theorem

Let $x_0, x_1, \dots, x_n \in [a, b]$ be distinct nodes and $f \in C^{2n+2}[a, b]$. If $p \in \Pi_{2n+1}$ such that $p(x_i) = f(x_i)$ and $p'(x_i) = f'(x_i)$ ($0 \leq i \leq n$). Then, for any $x \in [a, b]$, there is $\xi \in (a, b)$ such that

$$f(x) - p(x) = \frac{f^{(2n+2)}(\xi)}{(2n+2)!} \prod_{i=0}^n (x - x_i)^2.$$

Proof.

Skip. It is very similar to the Lagrange interpolation case. \square