Polynomial Interpolation

- Problem Formulation
- Divided Differences
- Interpolating Functions
- Hermite Interpolation

Contents

Polynomial Interpolation

We have $n+1$ data point $(x_0, y_0), \ldots, (x_n, y_n)$. We are interested in finding a polynomial

$$p(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n$$

such that $p(x_i) = y_i$ for all $i \in \{0, \ldots, n\}$.

Application Examples:

- We have a (smooth) function $f : \mathbb{R} \to \mathbb{R}$. Evaluating f at one point takes, say 1h. We need to evaluate f at 10^6 points $x \in [0, 1]$ within 6h. What can we do?
- We measure very accurately the lift force of a wing for 11 different angles of attack in $[0°, 10°]$. We want to predict the lift force of the wing at intermediate angles (but have no physical model at hand).

Polynomial Interpolation

We have $n+1$ data point $(x_0, y_0), \ldots, (x_n, y_n)$. We are interested in finding a polynomial

$$p(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n$$

such that $p(x_i) = y_i$ for all $i \in \{0, \ldots, n\}$.

Application Examples:

- We have a (smooth) function $f : \mathbb{R} \to \mathbb{R}$. Evaluating f at one point takes, say 1h. We need to evaluate f at 10^6 points $x \in [0, 1]$ within 6h. What can we do?
- We measure very accurately the lift force of a wing for 11 different angles of attack in $[0^\circ, 10^\circ]$. We want to predict the lift force of the wing at intermediate angles (but have no physical model at hand).

Existence and Uniqueness of Solutions

Theorem If none of the points $x_0, \ldots, x_n$ are equal, there exists a unique sequence of coefficients $a_0, \ldots, a_n$ such that $p(x_i) = y_i$ for all $i \in \{0, \ldots, n\}$, where

$$p(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n \ .$$

Proof: The proof of this theorem proceeds in two parts:

- *Existence:* construct a polynomial satisfying all requirements,
- *Uniqueness:* prove that if we have two interpolating polynomials, then they are equal.

Existence and Uniqueness of Solutions

Theorem If none of the points $x_0, \ldots, x_n$ are equal, there exists a unique sequence of coefficients $a_0, \ldots, a_n$ such that $p(x_i) = y_i$ for all $i \in \{0, \ldots, n\}$, where

$$p(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n \ .$$

Proof: The proof of this theorem proceeds in two parts:

- *Existence:* construct a polynomial satisfying all requirements,
- *Uniqueness:* prove that if we have two interpolating polynomials, then they are equal.

Existence and Uniqueness of Solutions

Theorem If none of the points $x_0, \ldots, x_n$ are equal, there exists a unique sequence of coefficients $a_0, \ldots, a_n$ such that $p(x_i) = y_i$ for all $i \in \{0, \ldots, n\}$, where

$$p(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n \ .$$

Proof: The proof of this theorem proceeds in two parts:

- *Existence:* construct a polynomial satisfying all requirements,
- *Uniqueness:* prove that if we have two interpolating polynomials, then they are equal.

Lagrange Polynomials

Lagrange's idea is to define auxiliary polynomials of the form

$$L_i(x) := \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j}$$

for all $i \in \{0, \ldots, n\}$.

Important Property:

$$L_i(x_k) = \left\{ \begin{array}{ll} 1 & \text{if } i = k \\ 0 & \text{otherwise} \end{array} \right\} = \delta_{i,k}$$

Thus, $p(x) = \sum_{k=0}^{n} y_k L_k(x)$ satisfies $p(x_i) = y_i$ for all $i \in \{0, \ldots, n\}$.

Lagrange Polynomials

Lagrange's idea is to define auxiliary polynomials of the form

$$L_i(x) := \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j}$$

for all $i \in \{0, \ldots, n\}$.

Important Property:

$$L_i(x_k) = \left\{ \begin{array}{ll} 1 & \text{if } i = k \\ 0 & \text{otherwise} \end{array} \right\} = \delta_{i,k}$$

Thus, $p(x) = \sum_{k=0}^{n} y_k L_k(x)$ satisfies $p(x_i) = y_i$ for all $i \in \{0, \ldots, n\}$.

Lagrange Polynomials

Lagrange's idea is to define auxiliary polynomials of the form

$$L_i(x) := \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j}$$

for all $i \in \{0, \ldots, n\}$.

Important Property:

$$L_i(x_k) = \left\{ \begin{array}{ll} 1 & \text{if } i = k \\ 0 & \text{otherwise} \end{array} \right\} = \delta_{i,k}$$

Thus, $p(x) = \sum_{k=0}^{n} y_k L_k(x)$ satisfies $p(x_i) = y_i$ for all $i \in \{0, \ldots, n\}$.

Example: Linear Interpolation

For $n = 1$ the problem reduces to finding a line ($=$ a polynomial with degree 1) passing through two given points

$$(x_0, y_0) \quad \text{and} \quad (x_1, y_1)\ .$$

The corresponding Lagrange polynomials are

$$L_0(x) = \frac{x - x_1}{x_0 - x_1} \quad \text{and} \quad L_1(x) = \frac{x - x_0}{x_1 - x_0}$$

Thus, the affine given function passing through the points is

$$\begin{aligned} p(x) &= y_0 L_0(x) + y_1 L_1(x) = y_0 \frac{x - x_1}{x_0 - x_1} + y_1 \frac{x - x_0}{x_1 - x_0} \\ &= \underbrace{\frac{y_0 - y_1}{x_0 - x_1}}_{a_1} x + \underbrace{\frac{y_1 x_0 - y_0 x_1}{x_0 - x_1}}_{a_0} = a_0 + a_1 x \end{aligned}$$

The function p satisfies $p(x_0) = y_0$ and $p(x_1) = y_1$.

Example: Linear Interpolation

For $n = 1$ the problem reduces to finding a line ($=$ a polynomial with degree 1) passing through two given points

$$(x_0, y_0) \quad \text{and} \quad (x_1, y_1)\ .$$

The corresponding Lagrange polynomials are

$$L_0(x) = \frac{x - x_1}{x_0 - x_1} \quad \text{and} \quad L_1(x) = \frac{x - x_0}{x_1 - x_0}$$

Thus, the affine given function passing through the points is

$$\begin{aligned} p(x) &= y_0 L_0(x) + y_1 L_1(x) = y_0 \frac{x - x_1}{x_0 - x_1} + y_1 \frac{x - x_0}{x_1 - x_0} \\ &= \underbrace{\frac{y_0 - y_1}{x_0 - x_1}}_{a_1} x + \underbrace{\frac{y_1 x_0 - y_0 x_1}{x_0 - x_1}}_{a_0} = a_0 + a_1 x \end{aligned}$$

The function p satisfies $p(x_0) = y_0$ and $p(x_1) = y_1$.

Example: Linear Interpolation

For $n = 1$ the problem reduces to finding a line ($=$ a polynomial with degree 1) passing through two given points

$$(x_0, y_0) \quad \text{and} \quad (x_1, y_1) \; .$$

The corresponding Lagrange polynomials are

$$L_0(x) = \frac{x - x_1}{x_0 - x_1} \quad \text{and} \quad L_1(x) = \frac{x - x_0}{x_1 - x_0}$$

Thus, the affine given function passing through the points is

$$\begin{aligned} p(x) &= y_0 L_0(x) + y_1 L_1(x) = y_0 \frac{x - x_1}{x_0 - x_1} + y_1 \frac{x - x_0}{x_1 - x_0} \\ &= \underbrace{\frac{y_0 - y_1}{x_0 - x_1}}_{a_1} x + \underbrace{\frac{y_1 x_0 - y_0 x_1}{x_0 - x_1}}_{a_0} = a_0 + a_1 x \end{aligned}$$

The function p satisfies $p(x_0) = y_0$ and $p(x_1) = y_1$.

Existence and Uniqueness of Solutions

Theorem If none of the points $x_0, \ldots, x_n$ are equal, there exists a unique sequence of coefficients $a_0, \ldots, a_n$ such that $p(x_i) = y_i$ for all $i \in \{0, \ldots, n\}$, where

$$p(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n \; .$$

Proof (Part I: *Existence*). The Lagrange polynomials

$$L_i(x) := \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j}$$

are well-defined (we never divide by zero), since $x_i \neq x_j$ for all $i \neq j$. The polynomial $p(x) = \sum_{k=0}^{n} y_k L_k(x)$ satisfies the requirments; that is, we have found (at least one) solution for p that is guaranteed to exist.

Existence and Uniqueness of Solutions

Theorem If none of the points $x_0, \ldots, x_n$ are equal, there exists a unique sequence of coefficients $a_0, \ldots, a_n$ such that $p(x_i) = y_i$ for all $i \in \{0, \ldots, n\}$, where

$$p(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n \ .$$

Proof (Part I: *Existence*). The Lagrange polynomials

$$L_i(x) := \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j}$$

are well-defined (we never divide by zero), since $x_i \neq x_j$ for all $i \neq j$. The polynomial $p(x) = \sum_{k=0}^{n} y_k L_k(x)$ satisfies the requirments; that is, we have found (at least one) solution for p that is guaranteed to exist.

Existence and Uniqueness of Solutions

Theorem If none of the points $x_0, \ldots, x_n$ are equal, there exists a unique sequence of coefficients $a_0, \ldots, a_n$ such that $p(x_i) = y_i$ for all $i \in \{0, \ldots, n\}$, where

$$p(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n \; .$$

Proof (Part II: *Uniqueness*). Assume that we can find two polynomials p, q with degree $\leq n$ which satisfy $p(x_i) = q(x_i) = y_i$ for $i \in \{0, \ldots, n\}$. The function $r(x) = p(x) - q(x)$ satisfies

$$r(x_i) = 0 \quad \text{for all} \quad i \in \{0, \ldots, n\}. \quad (n+1 \text{ roots})$$

Thus, $r(x) = 0$, since r is a polynomial of degree $\leq n$, i.e., $p = q$.

Existence and Uniqueness of Solutions

Theorem If none of the points $x_0, \ldots, x_n$ are equal, there exists a unique sequence of coefficients $a_0, \ldots, a_n$ such that $p(x_i) = y_i$ for all $i \in \{0, \ldots, n\}$, where

$$p(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n \ .$$

Proof (Part II: *Uniqueness*). Assume that we can find two polynomials p, q with degree $\leq n$ which satisfy $p(x_i) = q(x_i) = y_i$ for $i \in \{0, \ldots, n\}$. The function $r(x) = p(x) - q(x)$ satisfies

$$r(x_i) = 0 \quad \text{for all} \quad i \in \{0, \ldots, n\}. \quad (n + 1 \text{ roots})$$

Thus, $r(x) = 0$, since r is a polynomial of degree $\leq n$, i.e., $p = q$.

Contents

Disadvantages of Lagrange Polynomials

In practice, Lagrange polynomials are almost never used for interpolation. The two main reasons are:

1. Evaluating the expression $p(x) = \sum_{i=0}^{n} y_i \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j}$ is often not well-conditioned, i.e., we have to expect large numerical errors.
2. Say, we have already a polynomial passing through n data points (x_0, y_0), ..., (x_{n-1}, y_{n-1}) but now get a new data point (x_n, y_n). If we use Lagrange polynomials for computing a polynomial that passes through all data points, we have to compute this new polynomial from scratch.

Disadvantages of Lagrange Polynomials

In practice, Lagrange polynomials are almost never used for interpolation. The two main reasons are:

1. Evaluating the expression $p(x) = \sum_{i=0}^{n} y_i \prod_{j=0, j \neq i}^{n} \frac{x - x_j}{x_i - x_j}$ is often not well-conditioned, i.e., we have to expect large numerical errors.
2. Say, we have already a polynomial passing through n data points (x_0, y_0), ..., (x_{n-1}, y_{n-1}) but now get a new data point (x_n, y_n). If we use Lagrange polynomials for computing a polynomial that passes through all data points, we have to compute this new polynomial from scratch.

Disadvantages of Lagrange Polynomials

In practice, Lagrange polynomials are almost never used for interpolation. The two main reasons are:

1. Evaluating the expression $p(x) = \sum_{i=0}^{n} y_i \prod_{j=0, j\neq i}^{n} \frac{x-x_j}{x_i-x_j}$ is often not well-conditioned, i.e., we have to expect large numerical errors.
2. Say, we have already a polynomial passing through n data points (x_0, y_0), ..., (x_{n-1}, y_{n-1}) but now get a new data point (x_n, y_n). If we use Lagrange polynomials for computing a polynomial that passes through all data points, we have to compute this new polynomial from scratch.

Newton Polynomials

Newton's basis polynomials are given by

$$N_i(x) := \prod_{j=0}^{i-1} (x - x_j) \; .$$

The coefficients of the interpolating polynomial $p(x) = \sum_{i=0}^{n} b_i N_i(x)$ can be found by solving the equation system

$$\begin{aligned}
y_0 = p(x_0) &= b_0 \\
y_1 = p(x_1) &= b_0 + b_1(x_1 - x_0) \\
&\vdots \\
y_n = p(x_n) &= b_0 + b_1(x_n - x_0) + \ldots + b_n(x_n - x_0)\ldots(x_n - x_{n-1})
\end{aligned}$$

recursively with respect to $b_0, b_1, \ldots, b_n$.

Newton Polynomials

Newton's basis polynomials are given by

$$N_i(x) := \prod_{j=0}^{i-1} (x - x_j) \; .$$

The coefficients of the interpolating polynomial $p(x) = \sum_{i=0}^{n} b_i N_i(x)$ can be found by solving the equation system

$$\begin{aligned}
y_0 = p(x_0) &= b_0 \\
y_1 = p(x_1) &= b_0 + b_1(x_1 - x_0) \\
&\vdots \\
y_n = p(x_n) &= b_0 + b_1(x_n - x_0) + \ldots + b_n(x_n - x_0) \ldots (x_n - x_{n-1})
\end{aligned}$$

recursively with respect to $b_0, b_1, \ldots, b_n$.

Neville's Recursion Idea

Neville suggested a numerically stable way to find the coefficients of the interpolating polynomial using Newton's basis.

Key observation: if $(x_0, y_0), \ldots, (x_n, y_n)$ are given data points and f and g functions that satisfy

- $f(x_i) = y_i$ for all $i \in \{0, \ldots, n-1\}$ and
- $g(x_i) = y_i$ for all $i \in \{1, \ldots, n\}$,

then we can construct the new "divided-difference" function

$$h(x) = f(x) + \frac{g(x) - f(x)}{x_n - x_0}(x - x_0) ,$$

which satisfies $h(x_i) = y_i$ for all $i \in \{0, \ldots, n\}$.

Neville's Recursion Idea

Neville suggested a numerically stable way to find the coefficients of the interpolating polynomial using Newton's basis.

Key observation: if $(x_0, y_0), \ldots, (x_n, y_n)$ are given data points and f and g functions that satisfy

- $f(x_i) = y_i$ for all $i \in \{0, \ldots, n-1\}$ and
- $g(x_i) = y_i$ for all $i \in \{1, \ldots, n\}$,

then we can construct the new "divided-difference" function

$$h(x) = f(x) + \frac{g(x) - f(x)}{x_n - x_0}(x - x_0) \; ,$$

which satisfies $h(x_i) = y_i$ for all $i \in \{0, \ldots, n\}$.

Neville's Recursion Idea

Neville suggested a numerically stable way to find the coefficients of the interpolating polynomial using Newton's basis.

Key observation: if $(x_0, y_0), \ldots, (x_n, y_n)$ are given data points and f and g functions that satisfy

- $f(x_i) = y_i$ for all $i \in \{0, \ldots, n-1\}$ and
- $g(x_i) = y_i$ for all $i \in \{1, \ldots, n\}$,

then we can construct the new "divided-difference" function

$$h(x) = f(x) + \frac{g(x) - f(x)}{x_n - x_0}(x - x_0) \ ,$$

which satisfies $h(x_i) = y_i$ for all $i \in \{0, \ldots, n\}$.

Neville's Recursion Idea: example for $n = 2$

In order to understand Neville's recursion, we consider the case $n = 2$.

- The constant functions $p_{0,0}(x) = y_0$, $p_{11}(x) = y_1$, and $p_{22}(x) = y_2$ interpolate the first, second, and third data point, respectively.
- The polynomial $p_{0,1}(x) = p_{0,0}(x) + \frac{p_{11}(x) - p_{00}(x)}{x_1 - x_0}(x - x_0)$ satisfies $p_{0,1}(x_i)) = y_i$ for $i \in \{0, 1\}$.
- The polynomial $p_{1,2}(x) = p_{1,1}(x) + \frac{p_{22}(x) - p_{11}(x)}{x_2 - x_1}(x - x_1)$ satisfies $p_{0,1}(x_i)) = y_i$ for $i \in \{1, 2\}$.
- The polynomial $p(x) = p_{0,1} + \frac{p_{12}(x) - p_{01}(x)}{x_2 - x_0}(x - x_0)$ satisfies $p(x_i) = y_i$ for $i \in \{0, 1, 2\}$.

Neville's Recursion Idea: example for $n = 2$

In order to understand Neville's recursion, we consider the case $n = 2$.

- The constant functions $p_{0,0}(x) = y_0$, $p_{11}(x) = y_1$, and $p_{22}(x) = y_2$ interpolate the first, second, and third data point, respectively.
- The polynomial $p_{0,1}(x) = p_{0,0}(x) + \frac{p_{11}(x) - p_{00}(x)}{x_1 - x_0}(x - x_0)$ satisfies $p_{0,1}(x_i)) = y_i$ for $i \in \{0, 1\}$.
- The polynomial $p_{1,2}(x) = p_{1,1}(x) + \frac{p_{22}(x) - p_{11}(x)}{x_2 - x_1}(x - x_1)$ satisfies $p_{0,1}(x_i)) = y_i$ for $i \in \{1, 2\}$.
- The polynomial $p(x) = p_{0,1} + \frac{p_{12}(x) - p_{01}(x)}{x_2 - x_0}(x - x_0)$ satisfies $p(x_i) = y_i$ for $i \in \{0, 1, 2\}$.

Neville's Recursion Idea: example for $n = 2$

In order to understand Neville's recursion, we consider the case $n = 2$.

- The constant functions $p_{0,0}(x) = y_0$, $p_{11}(x) = y_1$, and $p_{22}(x) = y_2$ interpolate the first, second, and third data point, respectively.
- The polynomial $p_{0,1}(x) = p_{0,0}(x) + \frac{p_{11}(x) - p_{00}(x)}{x_1 - x_0}(x - x_0)$ satisfies $p_{0,1}(x_i)) = y_i$ for $i \in \{0, 1\}$.
- The polynomial $p_{1,2}(x) = p_{1,1}(x) + \frac{p_{22}(x) - p_{11}(x)}{x_2 - x_1}(x - x_1)$ satisfies $p_{0,1}(x_i)) = y_i$ for $i \in \{1, 2\}$.
- The polynomial $p(x) = p_{0,1} + \frac{p_{12}(x) - p_{01}(x)}{x_2 - x_0}(x - x_0)$ satisfies $p(x_i) = y_i$ for $i \in \{0, 1, 2\}$.

Neville's Recursion Idea: example for $n = 2$

In order to understand Neville's recursion, we consider the case $n = 2$.

- The constant functions $p_{0,0}(x) = y_0$, $p_{11}(x) = y_1$, and $p_{22}(x) = y_2$ interpolate the first, second, and third data point, respectively.
- The polynomial $p_{0,1}(x) = p_{0,0}(x) + \frac{p_{11}(x) - p_{00}(x)}{x_1 - x_0}(x - x_0)$ satisfies $p_{0,1}(x_i)) = y_i$ for $i \in \{0, 1\}$.
- The polynomial $p_{1,2}(x) = p_{1,1}(x) + \frac{p_{22}(x) - p_{11}(x)}{x_2 - x_1}(x - x_1)$ satisfies $p_{0,1}(x_i)) = y_i$ for $i \in \{1, 2\}$.
- The polynomial $p(x) = p_{0,1} + \frac{p_{12}(x) - p_{01}(x)}{x_2 - x_0}(x - x_0)$ satisfies $p(x_i) = y_i$ for $i \in \{0, 1, 2\}$.

Divided Differences

In general, Neville's recursion is initialized with

$$p_{i,i}(x) = y_i \quad \text{f.a.} \quad i \in \{1, \ldots, n\}$$

and applies the recursion rule

$$p_{i,i+k}(x) = p_{i,i+k-1}(x) + \frac{p_{i+1,i+k}(x) - p_{i,i+k-1}(x)}{x_{i+k} - x_i}(x - x_i)$$

for $k \in \{1, \ldots, n-i\}$ to finally compute $p(x) = p_{0,n}(x)$. This formula can be used directly, if $p(x)$ should be evaluated at a given point x.

Divided Differences

In general, Neville's recursion is initialized with

$$p_{i,i}(x) = y_i \quad \text{f.a.} \quad i \in \{1, \ldots, n\}$$

and applies the recursion rule

$$p_{i,i+k}(x) = p_{i,i+k-1}(x) + \frac{p_{i+1,i+k}(x) - p_{i,i+k-1}(x)}{x_{i+k} - x_i}(x - x_i)$$

for $k \in \{1, \ldots, n-i\}$ to finally compute $p(x) = p_{0,n}(x)$. This formula can be used directly, if $p(x)$ should be evaluated at a given point x.

Divided Differences

The divided differences are defined by the recursion

$$d_{ii} = y_i \quad \text{and} \quad d_{i,i+k} = \frac{d_{i+1,i+k} - d_{i,i+k-1}}{x_{i+k} - x_i}$$

for $i \in \{0, \dots, n\}$ and $k \in \{0, \dots, n-i\}$.

Theorem The functions $p_{i,i+k}(x)$ can be written in the form

$$p_{i,i+k}(x) = d_{i,i} + d_{i,i+1}(x - x_i) + \ldots + d_{i,i+k}(x - x_i) \ldots (x - x_{i+k-1}) \,.$$

for $i \in \{0, \dots, n\}$ and $k \in \{0, \dots, n-i\}$.

Proof: We have

$p_{i,i+k}(x) = p_{i,i+k-1}(x) + d_{i,i+k}(x - x_i) \ldots (x - x_{i+k-1})$ by construction. The proof follows by induction over k.

Divided Differences

The divided differences are defined by the recursion

$$d_{ii} = y_i \quad \text{and} \quad d_{i,i+k} = \frac{d_{i+1,i+k} - d_{i,i+k-1}}{x_{i+k} - x_i}$$

for $i \in \{0, \ldots, n\}$ and $k \in \{0, \ldots, n-i\}$.

Theorem The functions $p_{i,i+k}(x)$ can be written in the form

$$p_{i,i+k}(x) = d_{i,i} + d_{i,i+1}(x - x_i) + \ldots + d_{i,i+k}(x - x_i) \ldots (x - x_{i+k-1})\ .$$

for $i \in \{0, \ldots, n\}$ and $k \in \{0, \ldots, n-i\}$.

Proof: We have

$p_{i,i+k}(x) = p_{i,i+k-1}(x) + d_{i,i+k}(x - x_i) \ldots (x - x_{i+k-1})$ by

construction. The proof follows by induction over k.

Divided Differences

The divided differences are defined by the recursion

$$d_{ii} = y_i \quad \text{and} \quad d_{i,i+k} = \frac{d_{i+1,i+k} - d_{i,i+k-1}}{x_{i+k} - x_i}$$

for $i \in \{0, \ldots, n\}$ and $k \in \{0, \ldots, n-i\}$.

Theorem The functions $p_{i,i+k}(x)$ can be written in the form

$$p_{i,i+k}(x) = d_{i,i} + d_{i,i+1}(x - x_i) + \ldots + d_{i,i+k}(x - x_i) \ldots (x - x_{i+k-1}) \; .$$

for $i \in \{0, \ldots, n\}$ and $k \in \{0, \ldots, n-i\}$.

Proof: We have

$p_{i,i+k}(x) = p_{i,i+k-1}(x) + d_{i,i+k}(x - x_i) \ldots (x - x_{i+k-1})$ by construction. The proof follows by induction over k.

Visualization of Divided Differences

Divided differences can be computed recursively as visualized in the table below:

$$\begin{array}{c|c|} x_0 & y_0 \end{array}$$

For one data point, the constant polynomial $p(x) = y_0$ is the solution.

Visualization of Divided Differences

Divided differences can be computed recursively as visualized in the table below:

$$
\begin{array}{c|c|}
x_0 & y_0 \\
x_1 & y_1
\end{array}
$$

Let's assume a second data point becomes available.

Visualization of Divided Differences

Divided differences can be computed recursively as visualized in the table below:

$$\begin{array}{c|c|c} x_0 & y_0 & d_{01} \\ x_1 & y_1 & \end{array}$$

We compute $d_{01} = \frac{y_1 - y_0}{x_1 - x_0}$ and find the interpolating polynomial

$$p(x) = y_0 + d_{01}(x - x_0)\ .$$

Visualization of Divided Differences

Divided differences can be computed recursively as visualized in the table below:

$$
\begin{array}{c|c|c}
x_0 & y_0 & d_{01} \\
x_1 & y_1 & \\
x_2 & y_2 &
\end{array}
$$

Once the third data point is available...

Visualization of Divided Differences

Divided differences can be computed recursively as visualized in the table below:

$$\begin{array}{c|c|c} x_0 & y_0 & d_{01} \\ x_1 & y_1 & d_{12} \\ x_2 & y_2 & \end{array}$$

... we compute $d_{12} = \frac{y_2 - y_1}{x_2 - x_1}$ and ...

Visualization of Divided Differences

Divided differences can be computed recursively as visualized in the table below:

$$
\begin{array}{c|c|cc}
x_0 & y_0 & d_{01} & d_{02} \\
x_1 & y_1 & d_{12} & \\
x_2 & y_2 & &
\end{array}
$$

... $d_{02} = \frac{d_{12}-d_{01}}{x_2-x_0}$. The interpolating polynomial is

$$p(x) = y_0 + d_{01}(x - x_0) + d_{02}(x - x_0)(x - x_1) \; .$$

Visualization of Divided Differences

Divided differences can be computed recursively as visualized in the table below:

$$
\begin{array}{c|c|cc}
x_0 & y_0 & d_{01} & d_{02} \\
x_1 & y_1 & d_{12} & \\
x_2 & y_2 & &
\end{array}
$$

... $d_{02} = \frac{d_{12} - d_{01}}{x_2 - x_0}$.The interpolating polynomial is

$$p(x) = y_0 + d_{01}(x - x_0) + d_{02}(x - x_0)(x - x_1) \ .$$

Visualization of Divided Differences

Divided differences can be computed recursively as visualized in the table below:

$$
\begin{array}{c|c|ccc}
x_0 & y_0 & d_{01} & d_{02} & \mathbf{d_{03}} \\
x_1 & y_1 & d_{12} & \mathbf{d_{13}} & \\
x_2 & y_2 & \mathbf{d_{23}} & & \\
x_3 & \mathbf{y_3} & & &
\end{array}
$$

In general, the complexity of adding one data point is $\mathbf{O}(n)$.

The complexity for computing all coefficients is $\mathbf{O}(n^2)$.

Visualization of Divided Differences

Divided differences can be computed recursively as visualized in the table below:

$$\begin{array}{c|c|ccc} x_0 & y_0 & d_{01} & d_{02} & \mathbf{d_{03}} \\ x_1 & y_1 & d_{12} & \mathbf{d_{13}} & \\ x_2 & y_2 & \mathbf{d_{23}} & & \\ x_3 & \mathbf{y_3} & & & \end{array}$$

In general, the complexity of adding one data point is $\mathbf{O}(n)$.

The complexity for computing all coefficients is $\mathbf{O}(n^2)$.

Visualization of Divided Differences

Divided differences can be computed recursively as visualized in the table below:

$$
\begin{array}{c|c|cccc}
\mathbf{x_0} & \mathbf{y_0} & \mathbf{d_{01}} & \mathbf{d_{02}} & \mathbf{d_{03}} & \mathbf{d_{04}} \\
\mathbf{x_1} & y_1 & d_{12} & d_{13} & \mathbf{d_{14}} & \\
\mathbf{x_2} & y_2 & d_{23} & \mathbf{d_{24}} & & \\
\mathbf{x_3} & y_3 & \mathbf{d_{34}} & & & \\
\mathbf{x_4} & \mathbf{y_4} & & & &
\end{array}
$$

If we are only interested in computing p_{0n} we don't have to store all coefficients. Thus, the memory requirement scales with $\mathbf{O}(n)$.

Visualization of Divided Differences

Divided differences can be computed recursively as visualized in the table below:

$$
\begin{array}{c|c|ccccc}
x_0 & y_0 & d_{01} & d_{02} & d_{03} & d_{04} & d_{05} \\
x_1 & y_1 & d_{12} & d_{13} & d_{14} & d_{15} & \\
x_2 & y_2 & d_{23} & d_{24} & d_{25} & & \\
x_3 & y_3 & d_{34} & d_{35} & & & \\
x_4 & y_4 & d_{45} & & & & \\
x_5 & y_5 & & & & &
\end{array}
$$

We can keep on refining the scheme whenever new data points are available; $p(x) = \sum_{i=0}^{n} d_{0i} N_i(x)$.

Evaluation based on Horner's Scheme

Once the divided differences are computed the polynomial $p(x) = \sum_{i=0}^{n} d_{0i} N_i(x)$ can be evaluated at any given point x based on Horner's algorithm:

$$\begin{aligned} b_n &= d_{0n} \\ b_k &= d_{0k} + (x - x_k) b_{k+1} \quad k = n-1, \ldots, 0 \\ p(x) &= b_0. \end{aligned}$$

Contents

Polynomial Approximation Error

Polynomial interpolation can be applied to any set of data points.

However, often we are interested in approximating functions, i.e., the data points are

$$y_i = f(x_i) \,, \quad i \in \{0, \ldots, n\} \,,$$

where f is a $(n+1)$-times continuously differentiable function.

The difference between f and the interpolating polynomial can in this case be bounded by

$$|f(x) - p(x)| \leq \frac{1}{(n+1)!} \frac{\partial^{n+1} f(\xi_x)}{\partial x^{n+1}} \prod_{j=0}^{n} (x - x_j)$$

for a $\xi_x \in [\min_i x_i, \max_i x_i]$.

(for a proof see, e.g., the Numerical Analysis book by Burden and Faires)

Polynomial Approximation Error

Polynomial interpolation can be applied to any set of data points.

However, often we are interested in approximating functions, i.e., the data points are

$$y_i = f(x_i)\ , \quad i \in \{0, \ldots, n\}\ ,$$

where f is a $(n+1)$-times continuously differentiable function.

The difference between f and the interpolating polynomial can in this case be bounded by

$$|f(x) - p(x)| \leq \frac{1}{(n+1)!} \frac{\partial^{n+1} f(\xi_x)}{\partial x^{n+1}} \prod_{j=0}^{n} (x - x_j)$$

for a $\xi_x \in [\min_i x_i, \max_i x_i]$.

(for a proof see, e.g., the Numerical Analysis book by Burden and Faires)

Polynomial Approximation Error

Example 1: For the function $f(x) = \sin(x)$ all derivatives are uniformly bounded by 1 on the interval $[\underline{x}, \overline{x}]$. Thus, we have

$$|f(x) - p(x)| \leq \frac{1}{(n+1)!} \prod_{j=1}^{n} (x - x_j) \leq \frac{1}{(n+1)!} \, [\overline{x} - \underline{x}]^n \; ,$$

which converges to 0 for $n \to \infty$ as long as $x_i \in [\underline{x}, \overline{x}]$.

Example 2: For the function $f(x) = \frac{1}{1+x^2}$ the n-th derivative satisfies

$$|f^{(n)}(x)| \approx 2^n n! \, \mathbf{O}(|x|^{-2-n})$$

Here, a uniform convergence of polynomial interpolation cannot be expected (see Homework 2 for details).

Polynomial Approximation Error

Example 1: For the function $f(x) = \sin(x)$ all derivatives are uniformly bounded by 1 on the interval $[\underline{x}, \overline{x}]$. Thus, we have

$$|f(x) - p(x)| \leq \frac{1}{(n+1)!} \prod_{j=1}^{n} (x - x_j) \leq \frac{1}{(n+1)!} \left[\overline{x} - \underline{x}\right]^n ,$$

which converges to 0 for $n \to \infty$ as long as $x_i \in [\underline{x}, \overline{x}]$.

Example 2: For the function $f(x) = \frac{1}{1+x^2}$ the n-th derivative satisfies

$$|f^{(n)}(x)| \approx 2^n n! \, \mathbf{O}(|x|^{-2-n})$$

Here, a uniform convergence of polynomial interpolation cannot be expected (see Homework 2 for details).

Contents

Numerical Differentiation Revisited

What happens if we interpolate the points $(x, f(x))$ and $(x+h, f(x+h))$ for very small $h > 0$?

The slope of the interpolating polynomial approximates $f'(x)$:

$$\begin{array}{c|c|c} x & f(x) & \frac{f(x+h)-f(x)}{h} \\ x+h & f(x+h) & \end{array}$$

For $h \to 0$ this divided difference table becomes

$$\begin{array}{c|c|c} x & f(x) & f'(x) \\ x & f(x) & \end{array}$$

Numerical Differentiation Revisited

What happens if we interpolate the points $(x, f(x))$ and $(x+h, f(x+h))$ for very small $h > 0$?

The slope of the interpolating polynomial approximates $f'(x)$:

$$\begin{array}{c|c|c} x & f(x) & \frac{f(x+h)-f(x)}{h} \\ x+h & f(x+h) & \end{array}$$

For $h \to 0$ this divided difference table becomes

$$\begin{array}{c|c|c} x & f(x) & f'(x) \\ x & f(x) & \end{array}$$

Numerical Differentiation Revisited

What happens if we interpolate the points $(x, f(x))$ and $(x+h, f(x+h))$ for very small $h > 0$?

The slope of the interpolating polynomial approximates $f'(x)$:

$$\begin{array}{c|c|c} x & f(x) & \frac{f(x+h)-f(x)}{h} \\ x+h & f(x+h) & \end{array}$$

For $h \to 0$ this divided difference table becomes

$$\begin{array}{c|c|c} x & f(x) & f'(x) \\ x & f(x) & \end{array}$$

Numerical Differentiation Revisited

The same principle can be used to approximate higher order derivatives, for example

$$\begin{array}{c|c|cc} x-h & f(x-h) & \frac{f(x)-f(x-h)}{h} & \frac{f(x-h)-2f(x)+f(x+h)}{2h^2} \\ x & f(x) & \frac{f(x+h)-f(x)}{h} & \\ x+h & f(x+h) & & \end{array}$$

For $h \to 0$ this divided difference table becomes

$$\begin{array}{c|c|cc} x & f(x) & f'(x) & \frac{1}{2}f''(x) \\ x & f(x) & f'(x) & \\ x & f(x) & & \end{array}$$

Numerical Differentiation Revisited

The same principle can be used to approximate higher order derivatives, for example

$$
\begin{array}{c|c|cc}
x-h & f(x-h) & \frac{f(x)-f(x-h)}{h} & \frac{f(x-h)-2f(x)+f(x+h)}{2h^2} \\
x & f(x) & \frac{f(x+h)-f(x)}{h} & \\
x+h & f(x+h) & &
\end{array}
$$

For $h \to 0$ this divided difference table becomes

$$
\begin{array}{c|c|cc}
x & f(x) & f'(x) & \frac{1}{2}f''(x) \\
x & f(x) & f'(x) & \\
x & f(x) & &
\end{array}
$$

Hermite Interpolation

Hermite's interpolation problem is to find a polynomial of degree $\sum_{i=0}^{n} m_i$ satisfying the condition

$$\frac{\partial^k p_i}{\partial x^k}(x_i) = y_i^k \ , \quad k \in \{0, \ldots, m_i - 1\}$$

for all $i \in \{0, \ldots, n\}$ and data $y_i^k \in \mathbb{R}$.

The solution polynomial can be found in analogy to the standard interpolation problem with the only difference that the points x_i are added m_i times to the divided difference table. The divided differences are then replaced with the corresponding derivative terms.

Hermite Interpolation

Hermite's interpolation problem is to find a polynomial of degree $\sum_{i=0}^{n} m_i$ satisfying the condition

$$\frac{\partial^k p_i}{\partial x^k}(x_i) = y_i^k \ , \quad k \in \{0, \ldots, m_i - 1\}$$

for all $i \in \{0, \ldots, n\}$ and data $y_i^k \in \mathbb{R}$.

The solution polynomial can be found in analogy to the standard interpolation problem with the only difference that the points x_i are added m_i times to the divided difference table. The divided differences are then replaced with the corresponding derivative terms.

Hermite Interpolation

Example: we want to find a polynomial of degree 3, which satisfies

$$p(a) = f(a)\ , p'(a) = f'(a)\ ,\ p(b) = f(b)\ ,\ \text{and } p'(b) = f'(b)$$

for given points a, b and a continuously differentiable function f. The corresponding divided difference table is

$$\begin{array}{c|c|cccc}
a & f(a) & f'(a) & \frac{f(b)-f(a)-f'(a)(b-a)}{(b-a)^2} & \frac{2(f(a)-f(b))+(f'(a)+f'(b))(b-a)}{(b-a)^3} \\
a & f(a) & \frac{f(b)-f(a)}{b-a} & \frac{f(a)-f(b)+f'(b)(b-a)}{(b-a)^2} & \\
b & f(b) & f'(b) & & \\
b & f(b) & & &
\end{array}$$

This table yields the solution polynomial w.r.t. the Newton basis.

Hermite Interpolation

Example: we want to find a polynomial of degree 3, which satisfies

$$p(a) = f(a)\ , p'(a) = f'(a)\ ,\ p(b) = f(b)\ ,\ \text{and}\ p'(b) = f'(b)$$

for given points a, b and a continuously differentiable function f. The corresponding divided difference table is

$$\begin{array}{c|c|cccc}
a & f(a) & f'(a) & \frac{f(b)-f(a)-f'(a)(b-a)}{(b-a)^2} & \frac{2(f(a)-f(b))+(f'(a)+f'(b))(b-a)}{(b-a)^3} \\
a & f(a) & \frac{f(b)-f(a)}{b-a} & \frac{f(a)-f(b)+f'(b)(b-a)}{(b-a)^2} & \\
b & f(b) & f'(b) & & \\
b & f(b) & & &
\end{array}$$

This table yields the solution polynomial w.r.t. the Newton basis.

Approximation Error of Hermite Interpolation

For the general Hermite interpolation, the difference between f and the interpolating polynomial p is bounded by

$$|f(x) - p(x)| \leq \frac{1}{(m+1)!} \frac{\partial^{m+1} f(\xi_x)}{\partial x^{m+1}} \prod_{j=1}^{n} (x - x_j)^{m_j}$$

for a $\xi_x \in [\min_i x_i, \max_i x_i]$ and $m = \sum_{i=0}^{n} m_i$.

(for a proof see, e.g., the Numerical Analysis book by Burden and Faires)

Summary

- There exists a unique polynomial of order n, which interpolates the data points $(x_0, y_0), \ldots, (x_n, y_n)$, if $x_i \neq x_j$ for all $i \neq j$.
- The polynomial is given by $p(x) = \sum_{i=0}^{n} y_i L_i(x)$, but this representation is not used in practice.
- We have discussed how to use divided differences for computing interpolating polynomials.
- If the derivatives of f are uniformly bounded, the polynomial interpolation converges to the exact function for $n \to \infty$.
- Hermite interpolation additionally interpolates derivatives.