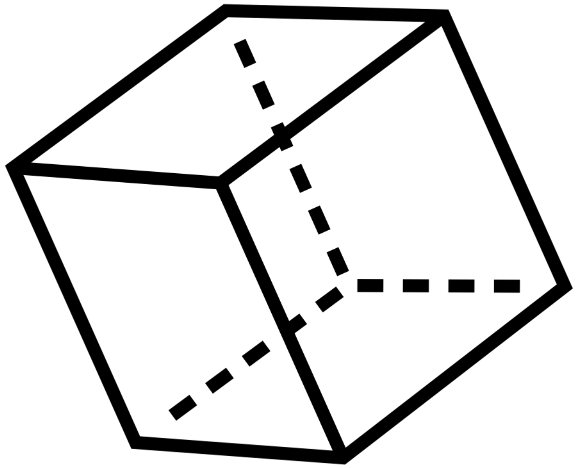


# Orthogonal Polynomials

# 2.1. Gram-Schmidt process

Question: How to appriximate the function $f$?

$$f \in C[a,b] \approx P_n(x) = \sum_{k=0}^{n} a_k x^k, \ \ E = \int_a^b (f - P_n)^2$$

Choose $a_k$ to minimize $E$: $(n+1)$ - normal equation

$$\Longrightarrow \sum_{k=0}^{n} a_k \left[ \int_a^b x^{k+j} dx \right] = \int_a^b f(x) x^j dx, \text{ where } j = 0, \cdots, n$$

# 2.1. Gram-Schmidt process

### Definition 2.1.1

An integrable function $w(x)$ is called a **weight function** on the interval $I$ if $w(x) \geq 0$ for all $x \in I$

### Example 2.1.1

$$w(x) = \frac{1}{\sqrt{1-x^2}} \text{ on } [-1, 1] \text{ is weight function.}$$

# 2.1. Gram-Schmidt process

## Definition 2.1.2

The set of functions $\{\phi_0, \cdots \phi_n\}$ is said to be **linearly independent** on $[a, b]$ if, whenever

$$c_0\phi_0(x) + c_1\phi_1(x) + \cdots + c_n\phi_n(x) = 0, \text{ for all } x \in [a, b]$$

we have $c_0 = c_1 = \cdots = c_n = 0$. Otherwise the set of functions is said to be **linearly dependent**.

## Theorem 2.1.1

Suppose that, for each $j = 0, 1, \cdots, n$, $\phi_j(x)$ is a polynomial of degree $j$. Then $\{\phi_0, \cdots, \phi_n\}$ is linearly independent on any interval $[a, b]$.

# 2.1. Gram-Schmidt process

Question: How to appriximate the function $f$?

$$f \in C[a,b] \approx P(x) = \sum_{k=0}^{n} a_k \phi_k, \;\; E = \int_a^b w(x) \Big[ f(x) - \sum_{k=0}^{n} a_k \phi_k(x) \Big]^2$$

for linearly independent set $\{\phi_0, \cdots, \phi_n\}$.

Choose $a_k$ to minimize $E$: $(n+1)$ - normal equation

$$\Longrightarrow \int_a^b w(x) f(x) \phi_j(x)\, dx = \sum_{k=0}^{n} a_k \int_a^b w(x) \phi_k(x) \phi_j(x)\, dx, \text{ where } j = 0, \cdots, n$$

# 2.1. Gram-Schmidt process

## Definition 2.1.3

$\{\phi_0, \cdots, \phi_n\}$ is said to be an **orthogonal set of functions** for the interval $[a, b]$ with respect to the weight function $w$ if

$$\int_a^b w(x)\phi_k(x)\phi_j(x)\,dx = \begin{cases} 0, & \text{when } j \neq k \\ \alpha_j > 0, & \text{when } j = k. \end{cases}$$

If, in addition, $\alpha_j = 1$ for each $j = 0, 1, \cdots, n$, the set is said to be **orthonormal**.

# 2.1. Gram-Schmidt process

## Theorem 2.1.2

If $\{\phi_0, \cdots, \phi_n\}$ is an orthogonal set of functions on an interval $[a, b]$ with respect to the weight function $w$, then the least squares approximation to $f$ on $[a, b]$ with respect to $w$ is

$$P(x) = \sum_{j=0}^{n} a_j \phi_j(x)$$

where, for each $j = 0,\ 1, \cdots, n$,

$$a_j = \frac{\int_a^b w(x)\phi_j(x) f(x)\, dx}{\int_a^b w(x)[\phi_j(x)]^2\, dx} = \frac{1}{\alpha_j} \int_a^b w(x)\phi_j(x) f(x)\, dx.$$

## 2.1. Gram-Schmidt process

### Theorem 2.1.3 (Gram-Schmidt process)

The set of polynomial functions $\{\phi_0, \cdots, \phi_n\}$ defined in the following way is orthogonal on $[a, b]$ with respect to the weight function $w$.

$$\phi_0(x) \equiv 1, \ \phi_1(x) = x - B_1, \text{ for each } x \text{ in } [a, b],$$

where

$$B_1 = \frac{\int_a^b x w(x)[\phi_0(x)]^2 \, dx}{\int_a^b w(x)[\phi_0(x)]^2 \, dx},$$

and when $k \geq 2$,

$$\phi_k(x) = (x - B_k)\phi_{k-1}(x) - C_k \phi_{k-2}(x), \text{ for each } x \text{ in } [a, b],$$

where

$$B_k = \frac{\int_a^b x w(x)[\phi_{k-1}(x)]^2 \, dx}{\int_a^b w(x)[\phi_{k-1}(x)]^2 \, dx},$$

and

$$C_k = \frac{\int_a^b x w(x)\phi_{k-1}(x)\phi_{k-2}(x) \, dx}{\int_a^b w(x)[\phi_{k-2}(x)]^2 \, dx},$$

## Example 2.1.2 (Legendre polynomials)

Legendre polynomial on $[-1, 1]$ with respect to $w(x) = 1$

Using the Gram-Schmidt process with $P_0(x) \equiv 1$ gives

$$B_1 = \frac{\int_{-1}^{1} x\,dx}{\int_{-1}^{1}\,dx} = 0,\ P_1(x) = x$$

Also,

$$B_2 = \frac{\int_{-1}^{1} x \cdot 1 \cdot x^2\,dx}{\int_{-1}^{1} x^2\,dx} = 0,\ C_2 = \frac{1}{3}$$

So,

$$P_2(x) = x^2 - \frac{1}{3}$$

# 2.2. Chebyshev Polynomials

The Chebyshev polynomials $\{T_n(x)\}$ are orthogonal on $(-1, 1)$ with respect to the weight function

$$w(x) = \frac{1}{\sqrt{1 - x^2}}.$$

For $x \in [-1, 1]$, define

$$T_n(x) = \cos[n \arccos x], \text{ for each } n \geq 0.$$

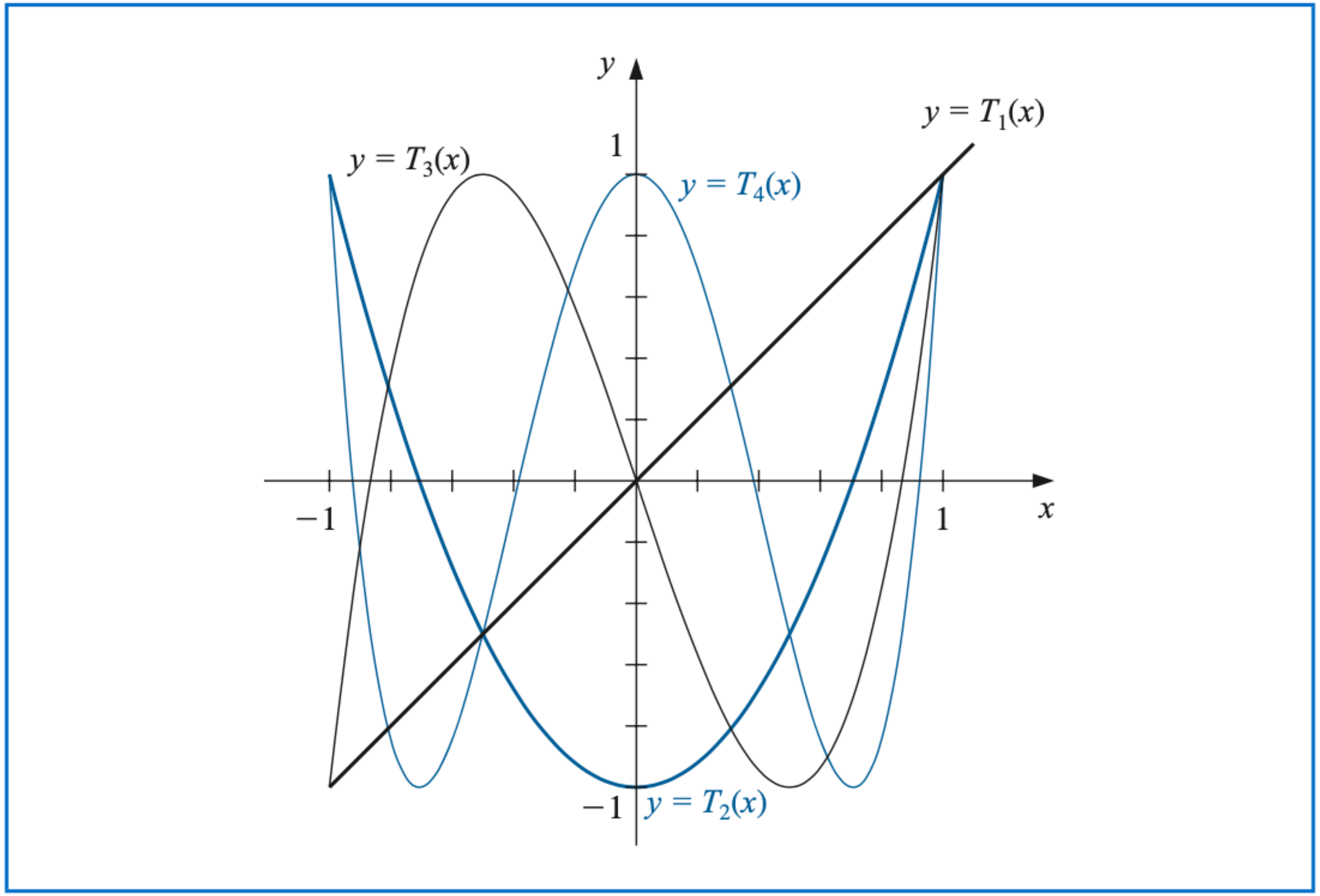
Note that

$$T_0(x) = \cos 0 = 1, \; T_1(x) = \cos(\arccos x) = x, \text{ and } T_{n+1} = 2xT_n(x) - T_{n-1}(x).$$

For example,

$$T_2(x) = 2x^2 - 1, \; T_3(x) = 4x^3 - 3x, \text{ and } T_4(x) = 8x^4 - 8x^2 + 1.$$

# 2.2. Chebyshev Polynomials



# 2.2. Chebyshev Polynomials

## Theorem 2.2.1

The Chebyshev polynomial $T_n(x)$ of degree $n \geq 1$ has $n$ simple zeros in $[-1, 1]$ at

$$\bar{x}_k = \cos\left(\frac{2k-1}{2n}\pi\right), \text{ for each } k = 1,\, 2, \cdots, n.$$

Moreover, $T_n(x)$ assumes its absolute extrema at

$$\bar{x}'_k = \cos\left(\frac{k\pi}{n}\right) \text{ with } T_n(\bar{x}'_k) = (-1)^k, \text{ for each } k = 0,\, 1, \cdots, n.$$

# 2.2. Chebyshev Polynomials

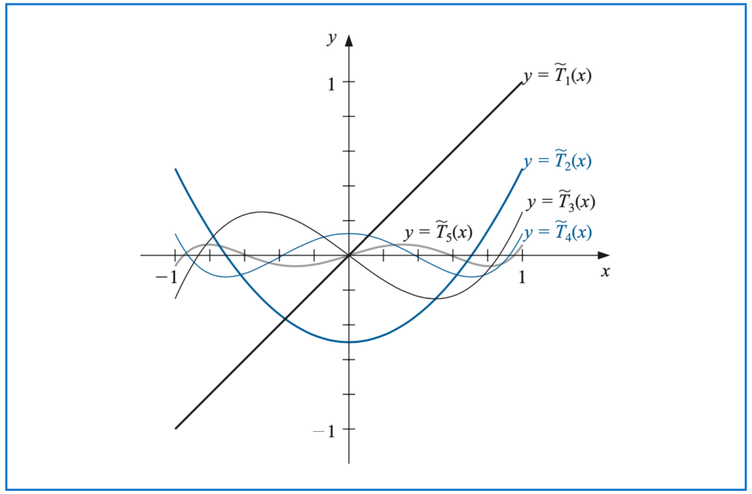
The monic (polynomials with leading coefficient 1) Chebyshev polynomials $\tilde{T}_n(x)$ are defined by

$$\tilde{T}_0(x) = 1 \text{ and } \tilde{T}_n(x) = \frac{1}{2^{n-1}} T_n(x), \text{ for each } n \geq 1.$$

The recurrence relationship satisfied by the Chebyshev polynomials implies that

$$\tilde{T}_2(x) = x\tilde{T}_1(x) - \frac{1}{2}\tilde{T}_0(x) \text{ and}$$

$$\tilde{T}_{n+1}(x) = x\tilde{T}_n(x) - \frac{1}{4}\tilde{T}_{n_1}(x), \text{ for each } n \geq 2.$$



# 2.2. Chebyshev Polynomials

The zeros of $\tilde{T}_n(x)$ also occur at

$$\tilde{x}_k = \cos\left(\frac{2k-1}{2n}\pi\right), \quad \text{for each } k = 1, 2, \cdots, n.$$

and the extreme values of $\tilde{T}_n(x)$, for $n \geq 1$, occur at

$$\tilde{x}'_k = \cos\left(\frac{k\pi}{n}\right), \quad \text{with } \tilde{T}_n(\tilde{x}'_k) = \frac{(-1)^k}{2^{n-1}}, \quad \text{for each } k = 0, 1, \cdots, n.$$

## Theorem 2.2.2

Let $\tilde{\Pi}_n$ denote the set of all monic polynomials of degree n. The polynomials of the form $\tilde{T}_n(x)$, when $n \geq 1$, have the property that

$$\frac{1}{2^{n-1}} = \max_{x\in[-1,1]} |\tilde{T}_n(x)| \leq \max_{x\in[-1,1]} |P_n(x)|, \quad \text{for all } P_n(x) \in \tilde{\Pi}_n$$

## 2.2. Chebyshev Polynomials

**Theorem 2.2.3 (Lagrange interpolation error)**

$$\text{For } f \in C^{n+1}(-1,1),\ |f(x)-P(x)| \le \frac{f^{(n+1)}(\xi(x))}{(n+1)!}(x-x_0)(x-x_1)\cdots(x-x_n).$$

Question: How to minimize the error with suitable $x_i$?

Since $(x-x_0)(x-x_1)\cdots(x-x_n)$ is a monic polynomial of degree $n+1$,
$\implies (x-x_0)\cdots(x-x_n) \in \tilde{\Pi}_{n+1}$.
By Thm 2.2.2, $\displaystyle\max_{x\in[-1,1]} |(x-x_0)\cdots(x-x_n)| \ge \frac{1}{2^n}$.

$\implies$ for $(x-x_0)\cdots(x-x_n) = \tilde{T}_{n+1}$, choose $x_k$ to be $\displaystyle \tilde{x}_{k+1} = \cos\left(\frac{2k+1}{2(n+1)}\pi\right),\ k=0,\cdots,n.$

$$\therefore \frac{1}{2^n} = \max_{x\in[-1,1]} |(x-\tilde{x}_1)\cdots(x-\tilde{x}_{n+1})| \le \max_{x\in[-1,1]} |(x-x_0)\cdots(x-x_n)|$$

# 2.2. Chebyshev Polynomials

Question: How to reduce the degree of approximating polynomials?

Let $P_n = \sum_{i=0}^{n} a_i x^i$ on $[-1,1]$. Choose $P_{n-1}(x)$ such that $\max_{x\in[-1,1]} |P_n(x) - P_{n-1}(x)|$ is as soon as possible.

Since the leading coefficient of $P_n(x) - P_{n-1}(x) = a_n$, $\frac{1}{a_n}(P_n(x) - P_{n-1}(x))$ is a monic polynomial.

By Thm 2.2.2, $\max_{x\in[-1,1]} \left| \frac{1}{a_n}(P_n(x) - P_{n-1}(x)) \right| \geq \frac{1}{2^{n-1}}$.

Since the equality hold for $\frac{1}{a_n}(P_n(x) - P_{n-1}(x)) = \tilde{T}_n(x)$, $P_{n-1}(x) = P_n(x) - a_n \tilde{T}_n(x)$.

$\therefore \max_{x\in[-1,1]} |P_n(x) - P_{n-1}(x)| = |a_n| \cdot \max_{x\in[-1,1]} |\tilde{T}_n(x)| = \frac{|a_n|}{2^{n-1}}$.

# 2.3. Trigonometric polynomial approximation

$$\phi_0 = \frac{1}{2},\ \phi_k(x) = \cos kx,\ k = 1, \cdots, n, \text{ and } \phi_{n+k}(x) = \sin kx,\ k = 1, \cdots, n-1.$$

Then $\{\phi_0, \cdots, \phi_{2n-1}\}$ is orthogonal set with respect to $w(x) = 1$ on $[-\pi, \pi]$.

For $f \in [-\pi, \pi]$, $S_n = \dfrac{a_0}{2} + a_n \cos nx + \displaystyle\sum_{k=1}^{n-1} (a_k \cos kx + b_k \sin kx)$.

By orthogonality, we have

$$a_k = \frac{\int_{-\pi}^{\pi} f(x) \cos kx \, dx}{\int_{-\pi}^{\pi} (\cos kx)^2 \, dx} = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \cos kx, \text{ and } b_k = \frac{\int_{-\pi}^{\pi} f(x) \sin kx \, dx}{\int_{-\pi}^{\pi} (\sin kx)^2 \, dx} = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \sin kx.$$

$\Longrightarrow \lim_{n \to \infty} S_n$ is Fourier series of $f(x)$.

## 2.3. Trigonometric polynomial approximation

### Lemma 2.3.1

Suppose that the integer $r$ is not a multiple of $2m$. Then

1) $\sum_{j=0}^{2m-1} \cos rx_j = 0$ and $\sum_{j=0}^{2m-1} \sin rx_j = 0$.

Moreover, if $r$ is not a multiple of $m$, then

2) $\sum_{j=0}^{2m-1} (\cos rx_j)^2 = m$ and $\sum_{j=0}^{2m-1} (\sin rx_j)^2 = m$, $x_j = -\pi + \left(\frac{j}{m}\right)\pi$ for each $j = 0, 1, \cdots, 2m-1$.

### Theorem 2.3.1

$S_n(x) = \frac{a_0}{2} + a_n \cos nx + \sum_{k=1}^{n-1}(a_k \cos kx + b_k \sin kx)$ minimizes the least square sum

$E(a_0, \cdots, a_n, b_1, \cdots, b_{n-1}) = \sum_{j=0}^{2m-1} (y_j - S_n(x_j))^2$ with

$a_k = \frac{1}{m} \sum_{j=0}^{2m-1} y_j \cos kx_j$, for $k = 0, \cdots, n$ and $b_k = \frac{1}{m} \sum_{j=0}^{2m-1} y_j \sin kx_j$, for $k = 0, \cdots, n-1$

# 2.4. Fast Fourier transforms

Cost of previous method: $O(2m)$ for all $k \implies O((2m)^2) \implies$ How to reduce it?

The fast Fourier transform procedure computes the complex coefficients $c_k$ in

$$\frac{1}{m}\sum_{k=0}^{2m-1} c_k e^{ikx},$$

where

$$c_k = \sum_{k=0}^{2m-1} y_j e^{ik\pi j/m} \text{ for each } k = 0,\, 1\cdots, 2m-1$$

Then, we can reduce the computing cost $O(m\log_2 m)$.

# Interpolation

# 3.1. Taylor expansion

**Theorem 3.1**

Suppose $f \in C^n[a,b]$, that $f^{(n+1)}$ exists on $[a,b]$, and $x_0 \in [a,b]$, there exists a number $\xi(x)$ between $x_0$ and $x$ with

$$f(x) = P_n(x) + R_n(x),$$

where

$$P_n(x) = f(x_0) + f'(x_0)(x-x_0) + \frac{f''(x_0)}{2!}(x-x_0)^2 + \cdots + \frac{f^{(n)}(x_0)}{n!}(x-x_0)^n = \sum_{k=0}^{n} \frac{f^{(k)}(x_0)}{k!}(x-x_0)^k$$

and

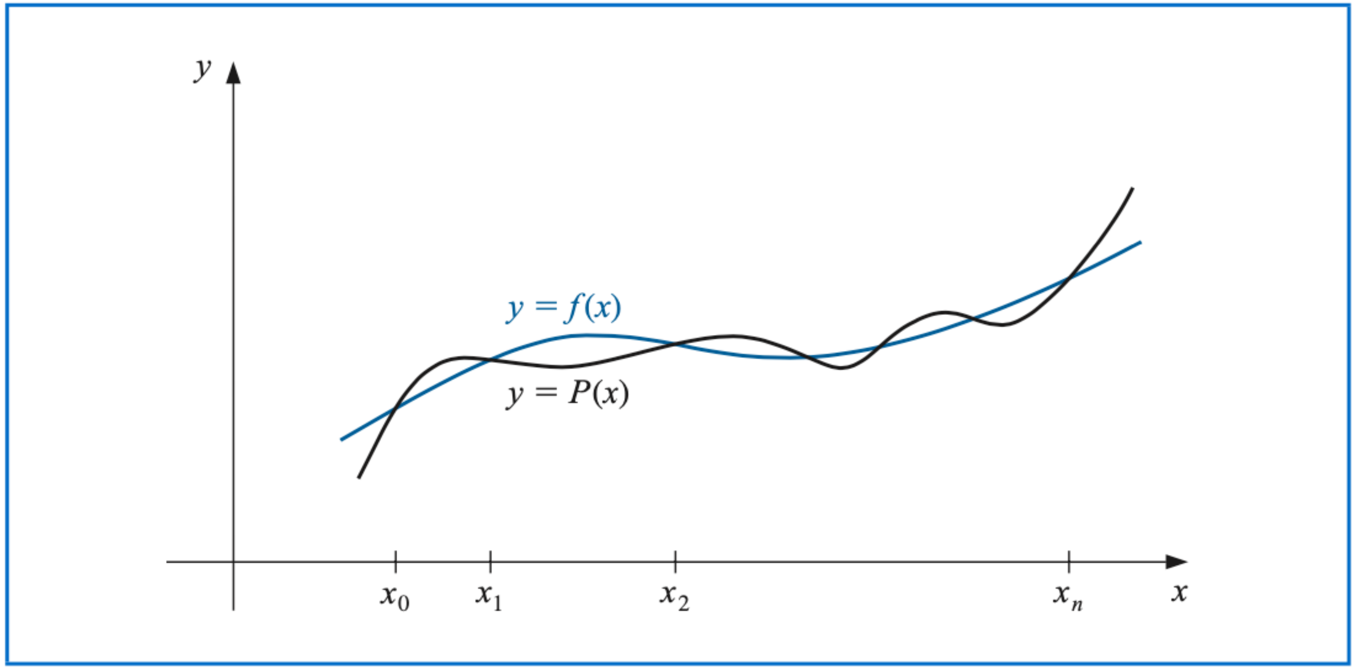
$$R_n(x) = \frac{f^{(n+1)}(\xi(x))}{(n+1)!}(x-x_0)^{n+1}.$$

# 3.2. Lagrange interpolation polynomial

**Theorem 3.2 (Weierstrass approximation theorem)**

Suppose that $f$ is defined and continuous on $[a, b]$. For each $\epsilon > 0$, there exists a polynomial $P(x)$, with the property that

$$|f(x) - P(x)| < \epsilon, \text{ for all } x \text{ in } [a, b]$$



# 3.2. Lagrange interpolation polynomial

## Theorem 3.3 (Lagrange interpolation polynomial)

If $x_0, x_1, \cdots, x_n$ are $n+1$ distinct numbers and $f$ is a function whose values are given at these numbers, then a unique polynomial $P(x)$ of degree at most $n$ exists with

$$f(x_k) = P(x_k), \text{ for each } k = 0, 1, \cdots, n.$$

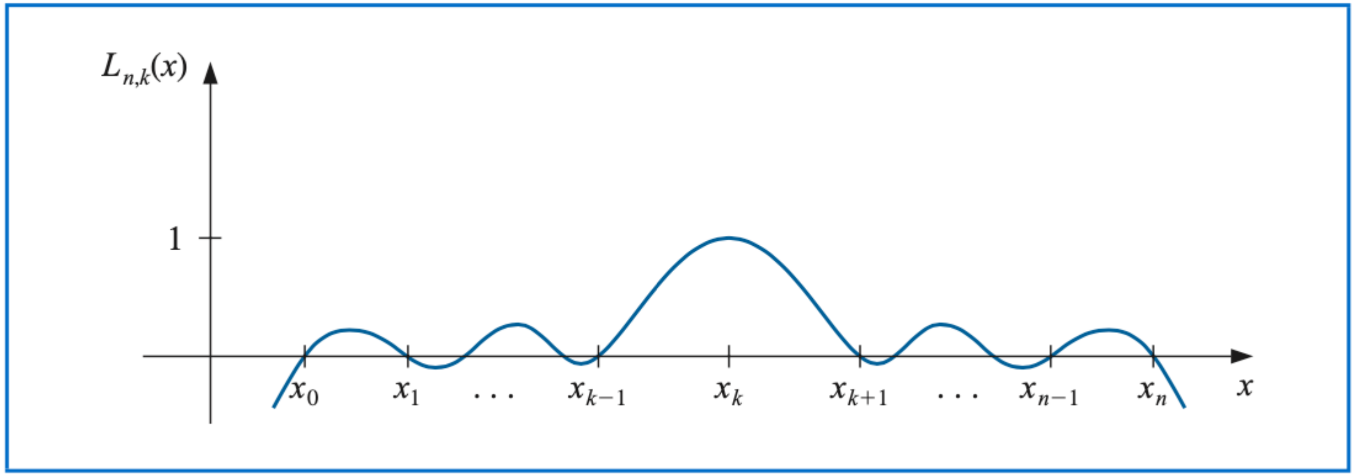
This polynomial is given by

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

where, for each $k = 0, 1, \cdots, n$,

$$L_{n,k} = \frac{(x-x_0)(x-x_1)\cdots(x-x_{k-1})(x-x_{k+1})\cdots(x-x_n)}{(x_k-x_0)(x_k-x_1)\cdots(x_k-x_{k-1})(x_k-x_{k+1})\cdots(x_k-x_n)} = \prod_{i=0, i\neq k}^{n} \frac{(x-x_i)}{(x_k-x_i)}.$$

# 3.2. Lagrange interpolation polynomial




**Theorem 3.4**

Suppose $x_0, x_1, \cdots, x_n$ are distinct numbers in the interval $[a, b]$ and $f \in C^{n+1}[a, b]$. Then, for each $x$ in $[a, b]$, a number $\xi(x)$ between $x_0, x_1, \cdots, x_n$, and hence in $(a, b)$, exist with

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

where $P(x)$ is the interpolating polynomial given in Thm 2.2.

# 3.4. Hermite interpolation

## Theorem 3.6

If $f \in C^1[a,b]$ and $x_0, \cdots, x_n \in [a,b]$ are distinct, the unique polynomial of least degree agreeing with $f$ and $f'$ at $x_0, \cdots, x_n$ is the Hermite polynomial of degree at most $2n+1$ given by

$$H_{2n+1}(x) = \sum_{j=0}^{n} f(x_j) H_{n,j}(x) + \sum_{j=0}^{n} f'(x_j) \hat{H}_{n,j}(x),$$

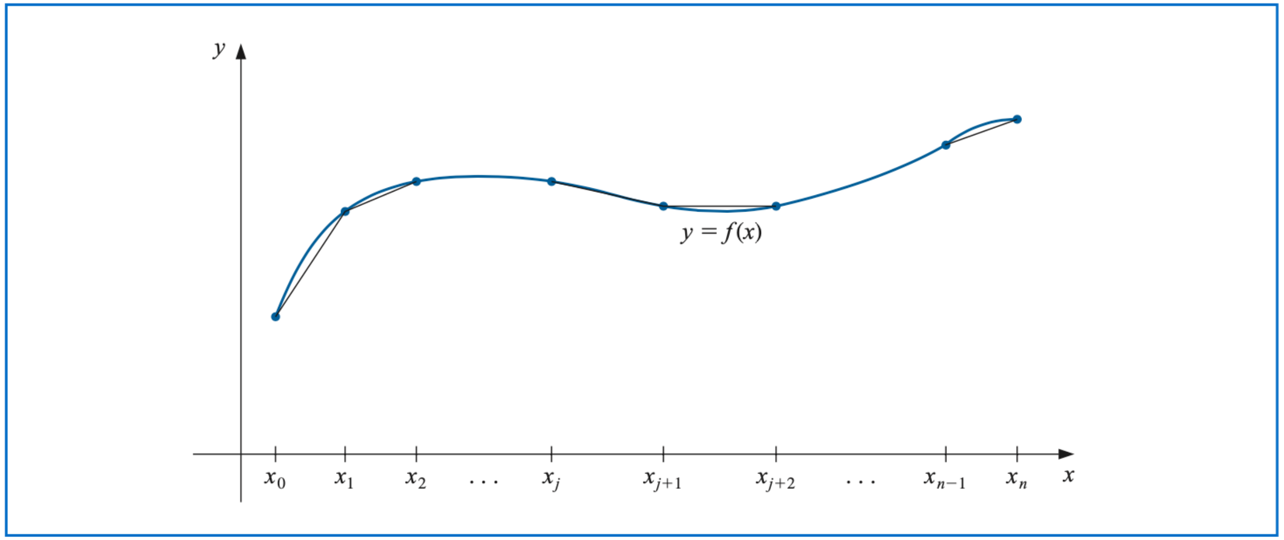
where, for $L_{n,j}(x)$ denoting the $j$th Lagrange coefficient polynomial of degree $n$, we have

$$H_{n,j}(x) = [1 - 2(x - x_j) L'_{n,j}(x_j)] L^2_{n,j}(x) \text{ and } \hat{H}_{n,j}(x) = (x - x_j) L^2_{n,j}(x).$$

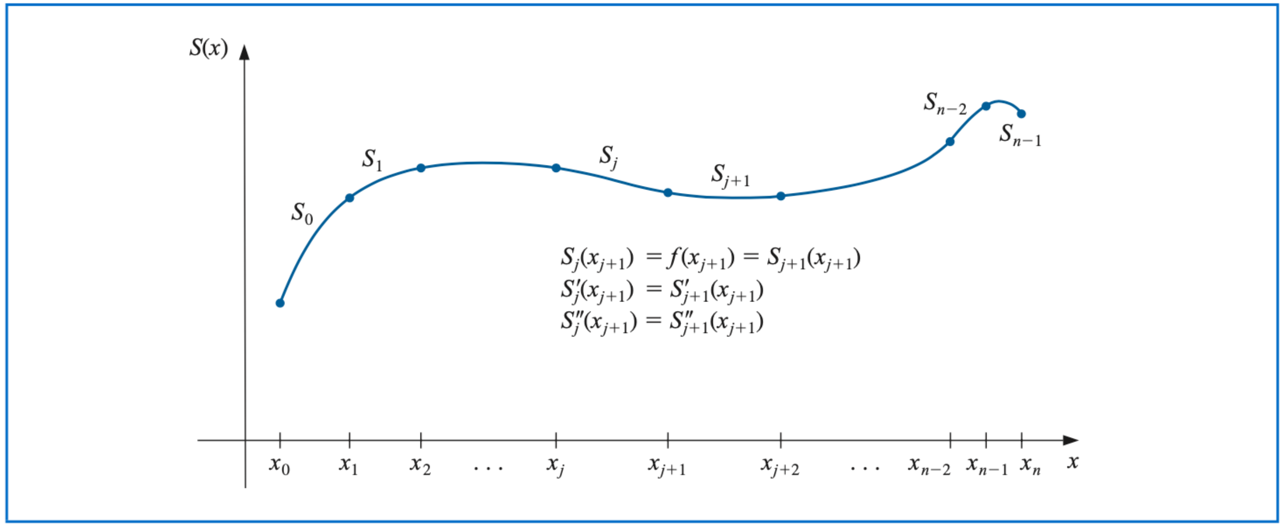
Moreover, if $f \in C^{2n+2}[a,b]$, then

$$f(x) = H_{2n+1}(x) + \frac{(x-x_0)^2 \cdots (x-x_n)^2}{(2n+2)!} f^{(2n+2)}(\xi(x))$$

# 3.5. Cubic spline interpolation




Piecewise-polynomial approximation




Piecewise cubic spline

# 3.5. Cubic spline interpolation

## Definition 3.1

Given a function $f$ is defined on $[a, b]$ and a set of nodes $a = x_0 < x_1 < \cdots < x_n = b$, a **cubic spline interpolant** $S$ for $f$ is a function that satisfies the following conditions:

1. $S(x)$ is a cubic polynomial, denoted $S_j(x)$, on the subinterval $[x_j, x_{j+1}]$ for each $j = 0, 1, \cdots, n-1$;
2. $S_j(x) = f(x_j)$ and $S_j(x_{j+1}) = f(x_{j+1})$ for each $j = 0, 1, \cdots, n-1$;
3. $S_{j+1}(x_{j+1}) = S_j(x_{j+1})$ for each $j = 0, 1, \cdots, n-2$;
4. $S'_{j+1}(x_{j+1}) = S'_j(x_{j+1})$ for each $j = 0, 1, \cdots, n-2$;
5. $S''_{j+1}(x_{j+1}) = S''_j(x_{j+1})$ for each $j = 0, 1, \cdots, n-2$;
6. One of the following sets of boundary conditions is satisfied:
   (a) $S''(x_0) = S''(x_n) = 0$ (natural or free boundary);
   (b) $S'(x_0) = f'(x_0)$ and $S'(x_n) = f'(x_n)$ (clamped boundary).

# 3.5. Cubic spline interpolation

Let $S_j(x) = a_j + b_j(x - x_j) + c_j(x - x_j)^2 + d_j(x - x_j)^3$ for each $j = 0,\ 1, \cdots, n$.
Since $S_j(x_j) = f(x_j)$,

$$a_j = f(x_j).$$

Define $a_n = f(x_n)$ then

$$a_j = f(x_j) \text{ for } j = 0,\ 1, \cdots, n.$$

Since $S_{j+1}(x_{j+1}) = S_j = (x_{j+1})$ and let $h_j = x_{j+1} - x_j$, then we can get

$$a_{j+1} = a_j + b_j h_j + c_j h_j^2 + d_j h_j^3,\ j = 0,\ 1, \cdots, n-1.$$

Since $S_j'(x) = b_j + 2c_j(x - x_j) + 3d_j(x - x_j)^2$,

$$S_j'(x_j) = b_j,\ j = 0,\ 1, \cdots, n.$$

Since $S_{j+1}'(x_{j+1}) = S_j'(x_{j+1})$,

$$b_{j+1} = b_j + 2c_j h_j + 3d_j h_j^2,\ j = 0,\ 1, \cdots, n-1.$$

# 3.5. Cubic spline interpolation

Since $S_j''(x) = 2c_j + 6d_j(x - x_j)$,

$$S_j''(x_j) = 2c_j,\ j = 0, \cdots, n-1 \implies c_j = \frac{S_j''(x_j)}{2}$$

Define $c_n = \dfrac{S_n''(x_n)}{2}$, then

$$c_j = \frac{S_j''(x_j)}{2},\ j = 0, \cdots, n$$

Since $S_{j+1}''(x_{j+1}) = S_j''(x_{j+1})$, $c_{j+1} = c_j + 3d_jh_j$, $j = 0, \cdots, n-1$,

$$\therefore d_j = \frac{(c_{j+1} - c_j)}{3h_j},\ j = 0, \cdots, n-1$$

# 3.5. Cubic spline interpolation

Since $a_{j+1} = a_j + b_j h_j + c_j h_j^2 + d_j h_j^3$, $j = 0, \cdots, n-1$,

$$a_{j+1} = a_j + b_j h_j + \frac{h_j^2}{3}(c_{j+1} + 2c_j) \tag{1}$$

Since $b_{j+1} = b_j + 2c_j h_j + 3d_j h_j^2$, $j = 0, \cdots, n-1$,

$$b_{j+1} = b_j + h_j(c_j + c_{j+1}) \tag{2}$$

Solving (1) for $b_j$, we have

$$b_j = \frac{1}{h_j}(a_{j+1} - a_j) - \frac{h_j}{3}(c_{j+1} + 2c_j) \tag{3}$$

From (2), we have

$$b_j = b_{j-1} + h_{j-1}(c_{j-1} + c_j) \tag{4}$$

# 3.5. Cubic spline interpolation

Substituting (3) to (4), we can get

$$h_{j-1}c_{j-1} + 2(h_{j-1} + h_j)c_j + h_j c_{j+1} = \frac{3}{h_j}(a_{j+1} - a_j) - \frac{3}{h_{j-1}}(a_j - a_{j-1})$$

Let $g_j = \frac{3}{h_j}(a_{j+1} - a_j) - \frac{3}{h_{j-1}}(a_j - a_{j-1})$, $j = 0, \cdots, n-1$ and choose $c_0 = 0 = c_n$ then we get,

$$\begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 & 0 & 1 \\ h_0 & 2(h_0 + h_1) & h_1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & h_1 & 2(h_1 + h_2) & h_2 & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \\ 0 & 0 & 0 & 0 & \cdots & h_{n-1} & 2(h_{n-1} + h_n) & h_n \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{n-1} \\ c_n \end{bmatrix} = \begin{bmatrix} 0 \\ g_1 \\ g_2 \\ \vdots \\ g_{n-1} \\ 0 \end{bmatrix}$$

It can be solved by the Thomas algorithm.

# 3.5. Cubic spline interpolation

⟨Thomas algorithm⟩

$$\begin{pmatrix} b_1 & c_1 & 0 & 0 \\ a_2 & b_2 & c_2 & 0 \\ 0 & a_3 & b_3 & c_3 \\ 0 & 0 & a_4 & b_4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \end{pmatrix}$$

Row 1: $b_1 x_1 + c_1 x_2 = r_1 \implies x_1 + \dfrac{c_1}{b_1} x_2 = \dfrac{r_1}{b_1}$

Choose $\dfrac{c_1}{b_1} = \gamma_1$ and $\dfrac{r_1}{b_1} = \rho_1$, we can get

$$x_1 + \gamma_1 x_2 = \rho_1 \tag{1}$$

Row 2: $a_2 x_1 + b_2 x_2 + c_2 x_3 = r_2$

Substituting (1) in this equation we can get $x_2 + \dfrac{c_2}{b_2 - a_2 x_1} x_3 = \dfrac{r_2 - a_2 \rho_1}{b_2 - a_2 \gamma_1}$

Choose $\dfrac{c_2}{b_2 - a_2 x_1} = \gamma_2$, $\dfrac{r_2 - a_2 \rho_1}{b_2 - a_2 \gamma_1} = \rho_2$, we can get

$$x_2 + \gamma_2 x_3 = \rho_2 \tag{2}$$

# 3.5. Cubic spline interpolation

⟨Thomas algorithm⟩

$$\begin{pmatrix} b_1 & c_1 & 0 & 0 \\ a_2 & b_2 & c_2 & 0 \\ 0 & a_3 & b_3 & c_3 \\ 0 & 0 & a_4 & b_4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \end{pmatrix}$$

Row 3: $a_3x_2 + b_3x_3 + c_3x_4 = r_3$

Substituting (2) in this equation we can get $x_3 + \dfrac{c_3}{b_3 - a_3\gamma_2}x_4 = \dfrac{r_3 - a_3\rho_2}{b_3 - a_2\gamma_2}$

Choose $\dfrac{c_3}{b_3 - a_3\gamma_2} = \gamma_3$, $\dfrac{r_3 - a_3\rho_2}{b_3 - a_3\gamma_2} = \rho_3$, we can get

$$x_3 + \gamma_3 x_4 = \rho_3 \tag{3}$$

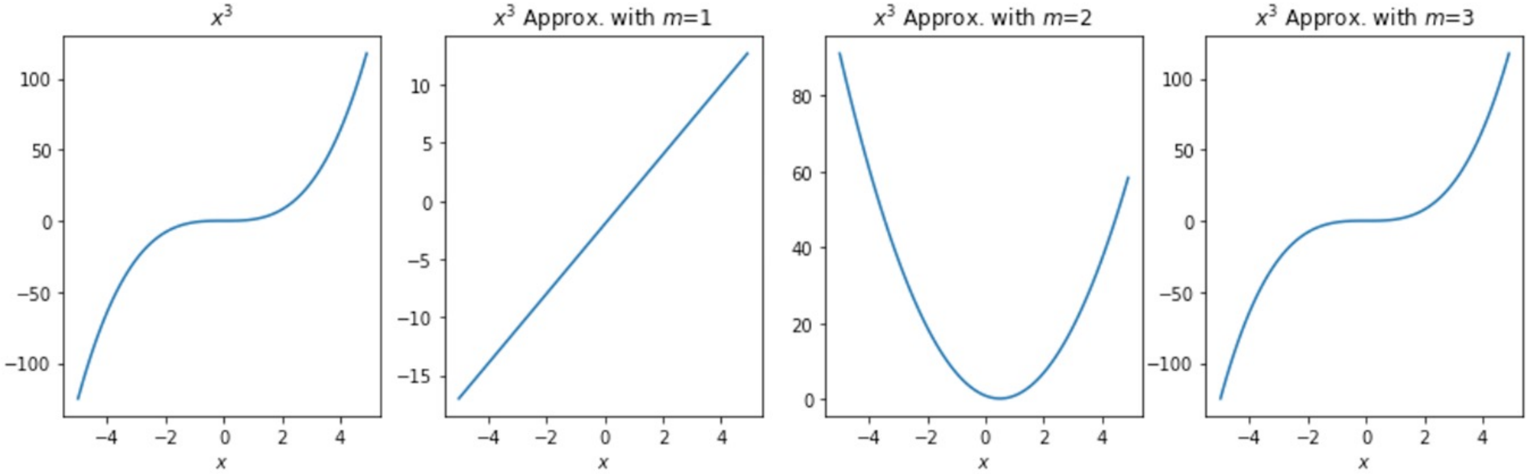
Row 4: $a_4x_3 + b_4x_4 = r_4$

Substituting (3) in this equation we can get

$$x_4 = \frac{r_4 - a_4\rho_3}{b_4 - a_4\gamma_3} = \rho_4 \tag{4}$$

# 3.6. Experiments

- **[노트북] 테일러 시리즈를 이용한 근사함수 계산**

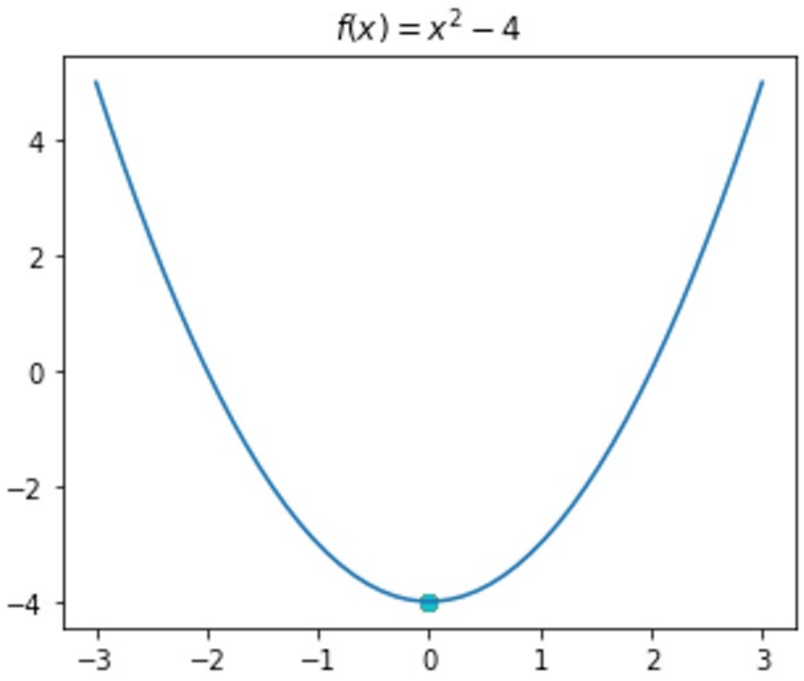



Google Colaboratory

https://colab.research.google.com/github/pp4e-book/pp4e-book....

# 3.6. Experiments

- **[노트북] scipy를 이용해 뉴턴법으로 최소값 찾기**




- **울산 지역 주택 가격 선형 회귀 예측 모델에서 최소값 찾기**
- **울산 토마토 가장 키우기 좋은 달 선형 회귀 예측 모델의 테일러 전개를 통한 근사 정도 시각화**

# 3.6. Experiments

**Table 3.18**

| $x$ | 0.9 | 1.3 | 1.9 | 2.1 | 2.6 | 3.0 | 3.9 | 4.4 | 4.7 | 5.0 | 6.0 | 7.0 | 8.0 | 9.2 | 10.5 | 11.3 | 11.6 | 12.0 | 12.6 | 13.0 | 13.3 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $f(x)$ | 1.3 | 1.5 | 1.85 | 2.1 | 2.6 | 2.7 | 2.4 | 2.15 | 2.05 | 2.1 | 2.25 | 2.3 | 2.25 | 1.95 | 1.4 | 0.9 | 0.7 | 0.6 | 0.5 | 0.4 | 0.25 |

**Figure 3.12**