Document for MTK

Zeyu Wang[1]

May 06, 2024

[1]Email: zeyu.wang.0117@outlook.com

Contents

Chapter 1

Polynomial

1.1 Single Variable Polynomial

Definition 1.1. Denoted by $\mathbb{V}$ a linear space and x the variable, a **(single variable) polynomial** over $\mathbb{V}$ is defined as

$$p_{n(x)} = \sum_{i=0}^{n} c_i x^i,$$

where $c_0, ..., c_n \in \mathbb{V}$ are constants that called the **coefficients of the polynomial**.

Definition 1.2. Given a polynomial $p(x) = \sum_{i=0}^{n} c_i x^i$ where $c_n \neq 0$, the degree of $p(x)$ is marked as $\deg(p(x)) = n$. In particular, the degree of zero polynomial $p(x) = 0$ is $\deg(0) = -\infty$.

Theorem 1.3. Denoted by $\mathbb{P}_n = \{p : \deg(p) \leq n\}$ the set of polynomials with degree no more than n ($n \geq 0$), and $\mathbb{P} = \bigcup_{n=0}^{\infty} \mathbb{P}_n$ the set contains all polynomials, then $\mathbb{P}_n$ is a linear space and satisfies

$$\{0\} = \mathbb{P}_0 \subset \mathbb{P}_1 \subset \cdots \subset \mathbb{P}_n \subset \cdots \mathbb{P}$$

1.2 Orthogonal Polynomial

Definition 1.4. Given a weight function $\rho(x) : [a, b] \to \mathbb{R}^+$, satisfies

$$\int_a^b \rho(x) \mathrm{d}x > 0, \int_a^b x^k \rho(x) \mathrm{d}x > 0 \quad \text{exists.}$$

The set of **orthogonal polynomials** on $[a, b]$ with the weight function $\rho(x)$ is defined as

$$\{p_i, i \in \mathbb{N}\} \subset L_\rho([a, b]) = \left\{ f(x) : \int_a^b f^2(x) \rho(x) \mathrm{d}x < \infty \right\}.$$

where $\{p_i, i \in \mathbb{N}\}$ are calculate from $\{x^n, n \in \mathbb{N}\}$ using the Gram-Schmidt process with the inner product

$$\forall f, g \in L_\rho([a, b]), \langle f, g \rangle = \int_a^b \rho(x) f(x) g(x) \mathrm{d}x.$$

Theorem 1.5. Orthogonal polynomials $p_{n-1}(x), p_n(x), p_{n+1}(x)$ satisfies

$$p_{n+1}(x) = (a_n + b_n x) p_n(x) + c_n p_{n-1}(x).$$

where a_n, b_n, c_n are depends on $[a, b]$ and ρ.

Theorem 1.6. The orthogonal polynomial $p_n(x)$ on $[a, b]$ with the weight function $\rho(x)$ has n roots on (a, b).

1.2.1 Legendre polynomial

Definition 1.7. The **Legendre polynomial** is defined on $[-1, 1]$ with the weight function $\rho(x) = 1$.

Theorem 1.8. The Legendre polynomials $\{p_i(x), i \in \mathbb{N}\}$ satisfies

$$\int_{-1}^{1} p_i(x)p_j(x)\mathrm{d}x = \begin{cases} \frac{2}{2i+1}, & i = j \\ 0, & i \neq j. \end{cases}$$

Theorem 1.9. The Legendre polynomial p_{n-1}, p_n, p_{n+1} satisfies

$$p_{n+1}(x) = \frac{2n+1}{n+1}xp_n(x) - \frac{n}{n+1}p_{n-1}(x).$$

Example 1.10. The first three terms of Legendre polynomials is

$$p_0(x) = 1, \quad p_1(x) = x, \quad p_2(x) = \frac{3}{2}x^2 - \frac{1}{2}.$$

1.2.2 Chebyshev polynomial of the first kind

Definition 1.11. The **Chebyshev polynomial of the first kind** is defined on $[-1, 1]$ with the weight function $\rho(x) = \frac{1}{\sqrt{1-x^2}}$.

Theorem 1.12. The Chebyshev polynomials of the first kind $\{p_i(x), i \in \mathbb{N}\}$ satisfies

$$\int_{-1}^{1} \frac{1}{\sqrt{1-x^2}} p_i(x)p_j(x)\mathrm{d}x = \begin{cases} \pi & i = j = 0 \\ \frac{\pi}{2} & i = j \neq 0 \\ 0 & i \neq j. \end{cases}$$

Theorem 1.13. The Chebyshev polynomial of the first kind p_{n-1}, p_n, p_{n+1} satisfies

$$p_{n+1}(x) = 2xp_n(x) - p_{n-1}(x).$$

Example 1.14. The first three terms of Chebyshev polynomials of the first kind is

$$p_0(x) = 1, \quad p_1(x) = x, \quad p_2(x) = 2x^2 - 1.$$

1.2.3 Chebyshev polynomial of the second kind

Definition 1.15. The **Chebyshev polynomial of the second kind** is defined on $[-1, 1]$ with the weight function $\rho(x) = \sqrt{1-x^2}$.

Theorem 1.16. The Chebyshev polynomials of the second kind $\{p_i(x), i \in \mathbb{N}\}$ satisfies

$$\int_{-1}^{1} \sqrt{1-x^2} p_i(x)p_j(x)\mathrm{d}x = \begin{cases} \frac{\pi}{2}, & i = j \\ 0, & i \neq j. \end{cases}$$

Theorem 1.17. The Chebyshev polynomial of the second kind p_{n-1}, p_n, p_{n+1} satisfies

$$p_{n+1}(x) = 2xp_n(x) - p_{n-1}(x).$$

Example 1.18. The first three terms of Chebyshev polynomials of the second kind is

$$p_0(x) = 1, \quad p_1(x) = 2x, \quad p_2(x) = 4x^2 - 1.$$

1.2.4 Laguerre polynomial

Definition 1.19. The **Laguerre polynomial** is defined on $[0, +\infty)$ with the weight function $\rho(x) = x^{\alpha}e^{-x}$.

Theorem 1.20. The Laguerre polynomial $\{p_i(x), i \in \mathbb{N}\}$ satisfies
$$\int_0^{+\infty} x^\alpha e^{-x} p_i(x) p_j(x) \mathrm{d}x = \begin{cases} \frac{\Gamma(n+\alpha+1)}{n!}, & i = j \\ 0, & i \neq j. \end{cases}$$

Theorem 1.21. For $\alpha = 0$, the Laguerre polynomial p_{n-1}, p_n, p_{n+1} satisfies
$$p_{n+1}(x) = (2n + 1 - x)p_n(x) - n^2 p_{n-1}(x).$$

Example 1.22. For $\alpha = 0$, the first three terms of Laguerre polynomial is
$$p_0(x) = 1, \quad p_1(x) = -x + 1, \quad p_2(x) = x^2 - 4x + 2.$$

1.2.5 Hermite polynomial (probability theory form)

Definition 1.23. The **Hermite polynomial** is defined on $(-\infty, +\infty)$ with the weight function $\rho(x) = \left(\frac{1}{\sqrt{2\pi}}\right) e^{-\frac{x^2}{2}}$.

Theorem 1.24. The Hermite polynomial $\{p_i(x), i \in \mathbb{N}\}$ satisfies
$$\int_0^{+\infty} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} p_i(x) p_j(x) \mathrm{d}x = \begin{cases} n!, & i = j \\ 0, & i \neq j. \end{cases}$$

Theorem 1.25. For $\alpha = 0$, the Hermite polynomial p_{n-1}, p_n, p_{n+1} satisfies
$$p_{n+1}(x) = x p_n(x) - n p_{n-1}(x).$$

Example 1.26. For $\alpha = 0$, the first three terms of Hermite polynomial is
$$p_0(x) = 1, \quad p_1(x) = x, \quad p_2(x) = x^2 - 1.$$

Chapter 2

Interpolation

2.1 Polynomial Interpolation

2.1.1 Lagrange formula

Definition 2.1. To interpolate given points $(x_0, f(x_0)), ..., (x_n, f(x_n))$, the Lagrange formula is

$$p_n(x) = \sum_{i=0}^{n} f(x_i) l_i(x),$$

where the **elementary Lagrange interpolation polynomial** (or **fundamental polynomial**) for pointwise interpolation $l_k(x)$ is

$$l_k(x) = \prod_{i=0}^{n} \frac{x - x_i}{x_k - x_i}.$$

In particular, for $n = 0, l_0(x) = 1$.

2.1.2 Newton formula

Definition 2.2. The kth divided difference ($k \in \mathbb{N}^+$) on the **table of divided differences**

$$\begin{array}{c|ccccc} x_0 & f[x_0] & & & \\ x_1 & f[x_1] & f[x_0, x_1] & & \\ x_2 & f[x_2] & f[x_1, x_2] & f[x_0, x_1, x_2] & \\ x_3 & f[x_3] & f[x_2, x_3] & f[x_1, x_2, x_3] & f[x_0, x_1, x_2, x_3] \\ \cdots & \cdots & \cdots & \cdots & \cdots \end{array}$$

where the **divided differences** satisfy

$$\begin{aligned} f[x_0] &= f(x_0), \\ f[x_0, ..., x_k] &= \frac{f[x_1, ..., x_k] - f\left[x_0, ..., x_{\{k-1\}}\right]}{x_k - x_0}. \end{aligned}$$

Corollary 2.3. Suppose $(i_0, ..., i_k)$ is a permutation of $(0, ..., k)$. Then

$$f[x_0, ..., x_k] = f\left[x_{i_0}, ..., x_{i_k}\right].$$

Theorem 2.4. For distinct points $x_0, ..., x_n$ and x, we have

$$f(x) = f[x_0] + f[x_0, x_1](x - x_0) + \cdots + f[x_0, ..., x_n] \prod_{i=0}^{n-1} (x - x_i) + f[x_0, ..., x_n, x] \prod_{i=0}^{n} (x - x_i).$$

Definition 2.5. The **Newton formula** for interpolating the points $(x_0, f(x_0)), ..., (x_n, f(x_n))$ is

$$p_n(x) = f[x_0] + f[x_0, x_1](x - x_0) + \cdots + f[x_0, ..., x_n] \prod_{i=0}^{n-1} (x - x_i).$$

2.1.3 Neville-Aitken algorithm

Definition 2.6. Denote $p_0^{[i]}(x) = f(x_i)$ for $i = 0, ..., n$. For all $k = 0, ..., n-1$ and $i = 0, ..., n-k-1$, define

$$p_{k+1}^{[i]}(x) = \frac{(x - x_i)p_k^{[i+1]}(x) - (x - x_{x+k+1})p_k^{[i]}(x)}{x_{i+k+1} - x_i}.$$

Then each $p_k^{[i]}(x)$ is the interpolating polynomial for the function f at the points $x_i, ..., x_{\{i+k\}}$. In particular, $p_n^{[0]}(x)$ is the interpolating polynomial of degree n for the function f at the points $x_0, ..., x_n$.

2.1.4 Hermite interpolation

Definition 2.7. Given distinct points $x_0, ..., x_k$ in $[a, b]$, non-negative integers $m_0, ..., m_k$, and a function $f \in C^M[a, b]$ where $M = \max_{i=0,...,k}(m_i)$, the **Hermite interpolation problem** seeks a polynomial $p(x)$ of the lowest degree satisfies

$$\forall i \in \{0, ..., k\}, \forall \mu \in \{0, ..., m_i\}, p^{(\mu)}(x_i) = f^{(\mu)}(x_i).$$

Definition 2.8. (Generalized divided difference) Let $x_0, ..., x_k$ be $k+1$ pairwise distinct points with each x_i repeated $m_i + 1$ times; write $N = k + \sum_{i=0}^{k} m_i$. The Nth divided difference associated with these points is the cofficient of x^N in the polynomial p that uniquely solves the Hermite interpolation problem.

Corollary 2.9. The nth divided difference at $n+1$ "confluent" (i.e. identical) points is

$$f[x_0, ..., x_0] = \frac{1}{n!} f^{(n)}(x_0),$$

where x_0 is repeated $n+1$ times on the left-hand side.

2.1.5 Approximation

Definition 2.10. Given condition functions $c_0, ..., c_k : \mathbb{P}_n \to \mathbb{R}^+$, the **Approximation problem** seeks a polynomial $p_n(x)$ of the given degree n satisfies a unconstrained optimization

$$\min_{p_n \in \mathbb{P}_n} \sum_{i=0}^{k} c_i\left(p_n^{(m_i)}\right).$$

where condition function $c(p)$ includes but is not limited to

$$|p^{(m)}(x)|, \left(p_n^{(m)}(x)\right)^2, \int_a^b |p^{(m)}| \, \mathrm{d}x, \int_a^b \left(p^{(m)}\right)^2 \mathrm{d}x.$$

Example 2.11. For non-negative integers $m_0, ..., m_k$ and condition functions $c_i(p_n) = \left(p_n^{(m_i)}(x)\right)^2$, denote by

$$p_n(x) = \sum_{i=0}^{n} c_i x^i$$

the polynomial of the given degree n, then the mth derivative of p_n is

$$p_n^{(m)}(x) = \sum_{i=m}^{n} \frac{i!}{(i-m)!} c_i x^{i-m}.$$

All above implies the least squares system

$$\begin{cases} p_n^{(m_0)}(x) = \sum_{i=m_0}^{n} \frac{i!}{(i-m_0)!} c_i x^{i-m_0} = 0, \\ \cdots \\ p_n^{(m_k)}(x) = \sum_{i=m_k}^{n} \frac{i!}{(i-m_k)!} c_i x^{i-m_k} = 0, \end{cases}$$

which can be solved by algorithms such as Householder transformation.

2.1.6 Error analysis

Theorem 2.12. Let $f \in C^n[a,b]$ and suppose that $f^{(n+1)}(x)$ exists at each point of (a,b). Let $p_n(x) \in \mathbb{P}_n$ denote the unique polynomial that coincides with f at $x_0, ..., x_n$. Define

$$R_n(f;x) = f(x) - p_n(x),$$

as the **Cauchy remainder** of the polynomial interpolation.
If $a \leq x_0 < \cdots < x_n \leq b$, then there exists some $\xi \in (a,b)$ satisfies

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

where the value of ξ depends on $x, x_0, ..., x_n$ and f.

Theorem 2.13. For the Hermite interpolation problem, denote $N = k + \sum_{i=0}^{k} m_i$. Denote by $p_N(x) \in \mathbb{P}_N$ the unique solution of the problem. Suppose $f^{(N+1)}(x)$ exists in (a,b). Then there exists some $\xi \in (a,b)$ satisfies

$$R_N(f;x) = \frac{f^{(N+1)}(\xi)}{(N+1)!} \prod_{i=0}^{k} (x - x_i)^{m_i+1}.$$

2.2 Spline

Definition 2.14. Given nonnegative integers n, k, and a strictly increasing sequence $a = x_1 < \cdots < x_N = b$, the set of **spline** functions of degree n and smoothness class k relative to the partition $\{x_i\}$ is

$$\mathbb{S}_n^k = \left\{ s : s \in C^k[a,b]; \forall i \in \{1, ..., N-1\}, s \mid_{[x_i, x_{i+1}]} \in \mathbb{P}_n \right\},$$

where x_i is the **knot** of the spline.

2.2.1 Cubic spline

Definition 2.15. (Boundary conditions of splines) The followings are common boundary conditions of cubic splines.

- The **complete cubic spline** s satisfies $s'(a) = f'(a), s'(b) = f'(b)$;
- The **cubic spline with specified second derivatives** s satisfies $s''(a) = f''(a), s''(b) = f''(b)$;
- The **natural cubic spline** s satisfies $s''(a) = s''(b) = 0$;
- The **not-a-knot cubic spline** s satisfies $s'''(x)$ exists at $x = x_2$ and $x = x_{N-1}$.
- The **periodic cubic spline** s satisfies $s(a) = s(b), s'(a) = s'(b), s''(a) = s''(b)$.

Theorem 2.16. Denote $m_i = s'(x_i), M_i = s''(x_i)$ for $s \in \mathbb{S}_3^2$, then

$$\forall i = 2, 3, ..., N-1, \quad \lambda_i m_{i-1} + 2m_i + \mu_i m_i + 1 = 3\mu_i f[x_i, x_{i+1}] + 3\lambda_i f[x_{i-1}, x_i],$$

$$\forall i = 2, 3, ..., N-1, \quad \mu_i M_{i-1} + 2M_i + \lambda_i m_{i+1} = 6f[x_{i-1}, x_i, x_{i+1}],$$

where

$$\mu_i = \frac{x_i - x_{i-1}}{x_{i+1} - x_{i-1}}, \quad \lambda_i = \frac{x_{i+1} - x_i}{x_{i+1} - x_{i-1}}.$$

In particular, m_i and M_i should be replaced to the derivatives given at the boundary.

Theorem 2.17. Cubic spline $s \in \mathbb{S}_3^2$ from the linear system of $\lambda_i, \mu_i, m_i, M_i$ and the boundary conditions.

2.2.2 B-spline

Definition 2.18. B-splines are defined recursively by

$$B_i^{n+1}(x) = (x - x_{i-1})(x_{i+n} - x_{i-1})B_i^n(x) + \frac{x_{i+n+1} - x}{x_{i+n+1} - x_{i-1}} B_{i+1}^n(x),$$

where recursion base is the B-spline of degree zero

$$B_i^0(x) = \begin{cases} 1, & x \in (x_{i-1}, x_i], \\ 0, & \text{otherwise.} \end{cases}$$

Theorem 2.19. The $\{B_i^n(x)\}$ forms a basis of $\mathbb{S}_n^{n-1}$.

Definition 2.20. For $N \in \mathbb{N}^*$, the **support** of a $B_i^n(x)$ is

$$\text{supp } \{B_i^n(x)\} = \overline{\{x \in \mathbb{R} : B_i^n(x) \neq 0\}} = [x_{i-1}, x_{i+n}].$$

Theorem 2.21. (Integrals of B-splines) The average of a B-spline over its support only depends on its degree,

$$\frac{1}{t_{i+n} - t_{i-1}} \int_{t_{i-1}}^{t_{i+n}} B_i^n(x) \mathrm{d}x = \frac{1}{n+1}.$$

Theorem 2.22. (Derivatives of B-splines) For $n \geq 2$, we have

$$\forall x \in \mathbb{R}, \quad \frac{\mathrm{d}}{\mathrm{d}x} B_i^n(x) = \frac{n B_i^{n-1}(x)}{x_{i+n-1} - x_{i-1}} - \frac{n B_{i+1}^{n-1}(x)}{x_{i+n} - x_i}.$$

For $n = 1$, it holds for all x except x_{i-1}, t_i, t_{i+1}, where the derivative of $B_i^1(x)$ is not defined.

2.2.3 Error analysis

Theorem 2.23. Suppose a function $f \in C^4[a, b]$, is interpolated by a complete cubic spline or a cubic spline with specified second derivatives at its end points. Then

$$\forall m = 0, 1, 2, |f^{(m)}(x) - s^{(m)}(x)| \leq c_m h^{4-m} \max_{x \in [a,b]} |f^{(4)}(x)|,$$

where $c_0 = \frac{1}{16}, c_1 = c_2 = \frac{1}{2}$ and $h = \max_{i=1,\ldots,N-1} |x_{i+1} - x_i|$.

Chapter 3

Integration

Definition 3.1. A **weighted quadrature formula** $I_n(f)$ is a linear function

$$I_n(f) = \sum_{i=1}^{n} w_i f(x_i),$$

which approximates the integral of a function $f \in C[a,b]$,

$$I(f) = \int_a^b \rho(x) f(x) \mathrm{d}x,$$

where the weight function $\rho \in [a,b]$ satisfies $\forall x \in (a,b),\ \rho(x) > 0$. The points $\{x_i\}$ at which the integrand f is evaluated are called nodes or abscissas, and the multipliers $\{w_i\}$ are called weights or coefficients.

Definition 3.2. A weighted quadrature formula has (polynomial) **degree of exactness** d_E iff

$$\forall f \in \mathbb{P}_{d_E}, \quad E_n(f) = 0,$$

$$\exists g \in \mathbb{P}_{d_E+1}, \ \text{s.t. } E_n(g) \neq 0$$

where $\mathbb{P}_d$ denotes the set of polynomials with degree no more than d.

Theorem 3.3. A weighted quadrature formula $I_n(f)$ satisfies $d_E \leq 2n - 1$.

Definition 3.4. The **error** or **remainder** of $I_n(f)$ is

$$E_n(f) = I(f) - I_n(f),$$

where $I_n(f)$ is said to be convergent for $C[a,b]$ iff

$$\forall f \in C[a,b], \lim_{n \to +\infty} E_n(f) = 0.$$

Theorem 3.5. Let $x_1, ..., x_n$ be given as distinct nodes of $I_n(f)$. If $d_E \geq n - 1$, then its weights can be deduced as

$$\forall k \in \{1, ..., n\}, w_k = \int_a^b \rho(x) l_k(x) \mathrm{d}x,$$

where $l_k(x)$ is the elementary Lagrange interpolation polynomial for pointwise interpolation applied to the given nodes.

3.1 Newton-Cotes Formulas

Definition 3.6. A **Newton-Cotes formula** is a formula based on approximating $f(x)$ by interpolating it on uniformly spaced nodes $x_1, ..., x_n \in [a,b]$.

3.1.1 Midpoint rule

Definition 3.7. The **midpoint rule** is a formula based on approximating $f(x)$ by the constant $f\left(\frac{a+b}{2}\right)$.
For $\rho(x) \equiv 1$, it is simply

$$I_M(f) = (b-a) f\left(\frac{a+b}{2}\right).$$

Theorem 3.8. For $f \in C^2[a,b]$, with weight functino $\rho \equiv 1$, the error (remainder) of midpoint rule satisfies

$$\exists \xi \in [a,b], \text{ s.t. } E_M(f) = \frac{(b-a)^3}{24} f''(\xi).$$

Corollary 3.9. The midpoint rule has $d_E = 1$.

3.1.2 Trapezoidal rule

Definition 3.10. The **trapezoidal rule** is a formula based on approximating $f(x)$ by the straight line that connects $(a, f(a))$ and $(b, f(b))$.
For $\rho(x) \equiv 1$, it is simply

$$I_T(f) = \frac{b-a}{2}(f(a) + f(b)).$$

Theorem 3.11. For $f \in C^2[a,b]$, with weight functino $\rho \equiv 1$, the error (remainder) of trapezoidal rule satisfies

$$\exists \xi \in [a,b], \text{ s.t. } E_T(f) = -\frac{(b-a)^3}{12} f''(\xi).$$

Corollary 3.12. The trapezoidal rule has $d_E = 1$.

3.1.3 Simpson's rule

Definition 3.13. The **Simpson's rule** is a formula based on approximating $f(x)$ by the quadratic polynomial that goes through the points $(a, f(a))$, $\left(\frac{a+b}{2}, f\left(\frac{a+b}{2}\right)\right)$ and $(b, f(b))$.
For $\rho(x) \equiv 1$, it is simply

$$I_S(f) = \frac{b-a}{6}\left(f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right).$$

Theorem 3.14. For $f \in C^4[a,b]$, with weight functino $\rho \equiv 1$, the error (remainder) of Simpson's rule satisfies

$$\exists \xi \in [a,b], \text{ s.t. } E_T(f) = -\frac{(b-a)^5}{2880} f^{(4)}(\xi).$$

Corollary 3.15. The Simpson's rule has $d_E = 3$.

3.2 Gauss Formulas

Theorem 3.16. For an interval $[a,b]$ and a weight function $\rho : [a,b] \to \mathbb{R}$, the nodes for gauss formula $I_n(f)$ is the root of the nth order orthogonal polynomial on $[a,b]$ with the weight function $\rho(x)$.

Theorem 3.17. A Gauss formula $I_n(f)$ has $d_E = 2n - 1$.

Chapter 4

Optimization

4.1 One-dimensional Line Search

Definition 4.1. Given a function $f : \mathbb{R}^n \to \mathbb{R}$, a initial point $\mathbf{x}$ and a direction $\mathbf{d}$, denoted by $\varphi(\alpha) = f(\mathbf{x} + \alpha\mathbf{d})$, a **one-dimensional line search** method solves the problem

$$\varphi(\alpha) = \min_{t\in\mathbb{R}^+} \varphi(t).$$

Method 4.2. (Success-failure method) For a one-dimensional line search problem, the **success-failure method** is an inexact one-dimensional line search method to solve the interval $[a, b] \in [0, +\infty)$ that exact solution $\alpha^* \in [a, b]$, where we

(1) Choose initial value $\alpha_0 \in [0, +\infty)$, $h_0 > 0$, $t > 0$(commonly choose $t = 2$), calculate $\varphi(\alpha_0)$ and let $k = 0$;

(2) Let $\alpha_{k+1} = \alpha_k + h_k$ and calculate $\varphi(\alpha_{k+1})$, if $\varphi(\alpha_{k+1}) < \varphi(\alpha_k)$, then go to (3), otherwise go to (4);

(3) Let $h_{k+1} = th_k$, $\alpha = \alpha_k$, $k = k + 1$, and go to (2);

(4) If $k = 0$, then let $h_k = -h_k$ and go to (2), otherwise stop and the solution $[a, b]$ satisfies

$$a = \min\{\alpha, \alpha_k\}, \quad b = \max\{\alpha, \alpha_k\}.$$

Definition 4.3. A general form of one-dimensional line search method is the following three steps:

(1) **Initialization**: given initial point $\mathbf{x}$ and acceptable error $\varepsilon > 0$, $\delta > 0$;

(2) **Iteration**: calculate the direction $\mathbf{d}$ and step size α that $f(\mathbf{x} + \alpha\mathbf{d}) = \min_{t\in\mathbb{R}^+} f(\mathbf{x} + t\mathbf{d})$ and let $\mathbf{x} = \mathbf{x} + \alpha\mathbf{d}$;

(3) **Stop condition**: if $\|\nabla f(\mathbf{x})\| \le \varepsilon$ or $U_{\mathbb{R}^n}(x, \delta)$ includes the exact solution, then the current $\mathbf{x}$ is the solution.

where the iteration step are repeated until $\mathbf{x}$ satisfies the stop condition.

Definition 4.4. Given a method, denoted by $\{\mathbf{x}_k\}$ the sequence of the iteration and $\mathbf{x}^*$ the exact solution, the method is **(Q-)linear convergence** if

$$\lim_{k\to\infty} \frac{\|\mathbf{x}_{k+1} - \mathbf{x}^*\|}{\|\mathbf{x}_k - \mathbf{x}^*\|} \in (0, 1),$$

the method is **(Q-)sublinear convergence** if

$$\lim_{k\to\infty} \frac{\|\mathbf{x}_{k+1} - \mathbf{x}^*\|}{\|\mathbf{x}_k - \mathbf{x}^*\|} = 1,$$

the method is **(Q-)superlinear convergence** if

$$\lim_{k\to\infty} \frac{\|\mathbf{x}_{k+1} - \mathbf{x}^*\|}{\|\mathbf{x}_k - \mathbf{x}^*\|} = 0.$$

For a superlinear convergence method, the method is r-order linear convergence if

$$\lim_{k\to\infty} \frac{\|\mathbf{x}_{k+1} - \mathbf{x}^*\|}{\|\mathbf{x}_k - \mathbf{x}^*\|^r} \in [0, +\infty),$$

where when $r = 2$ is called **(Q-)quadratic convergence**.

Remark 4.5. There is another **R-convergence** for judging a sequence which use another Q-convergence sequence as the boundary of $\{\|\mathbf{x}_k - x^*\|\}$, but is not needed here.

Method 4.6. (Golden section method) Given the initial point $\mathbf{x}$, an interval $[a, b]$ and $\delta > 0$,

- The iteration step is:
  (1) Calculate the two testing points $\lambda = a + (1 - k)(b - a)$ and $\mu = a + k(b - a)$ where $k = \frac{\sqrt{5}-1}{2}$ is the golden ratio;
  (2) If $\varphi(\lambda) > \varphi(\mu)$, let $a = \lambda$, otherwise let $b = \mu$.
- The stop condition is $b - a \leq \delta$;
- The solution is $\mathbf{x} + \frac{a+b}{2}\mathbf{d}$.

Theorem 4.7. The golden section method is a **linear convergent** method.

Method 4.8. (Fibonacci method) Given the initial point $\mathbf{x}$, an interval $[a, b]$ and $\delta > 0$,

- The k-th iteration step is:
  (1) Calculate the two testing points $\lambda = a + \frac{F_k}{F_{k+2}}(b - a)$ and $\mu = a + \frac{F_{k+1}}{F_{k+2}}(b - a)$ where F_k is the k-th fibonacci number and k;
  (2) If $\varphi(\lambda) > \varphi(\mu)$, let $a = \lambda$, otherwise let $b = \mu$.
- The stop condition is $b - a \leq \delta$;
- The solution is $\mathbf{x} + \frac{a+b}{2}\mathbf{d}$.

Theorem 4.9. The Fibonacci method is a **linear convergent** method.

Method 4.10. (Bisection method) Given the initial point $\mathbf{x}$, an interval $[a, b]$ and $\delta > 0$,

- The iteration step is:
  (1) Calculate the midpoint $m = \frac{a+b}{2}$ and $\varphi(m)$;
  (2) If $\nabla f(m) \cdot d < 0$, let $a = m$, otherwise let $b = m$.
- The stop condition is $b - a \leq \delta$;
- The solution is $\mathbf{x} + \frac{a+b}{2}\mathbf{d}$.

Theorem 4.11. The bisection method is a **linear convergent** method.

Method 4.12. (Newton's method) Given the initial point $\mathbf{x}$ and $\varepsilon > 0$,

- The iteration step is:
  (1) Calculate $\left(\nabla^2 f(\mathbf{x})\right)^T \cdot \mathbf{d}$ and $(\nabla f(\mathbf{x}))^T \cdot \mathbf{d}$;
  (2) Let $\mathbf{x} = \mathbf{x} - \frac{(\nabla f(\mathbf{x}))^T \cdot \mathbf{d}}{(\nabla^2 f(\mathbf{x}))^T \cdot \mathbf{d}}$;
- The stop condition is $(\nabla f(\mathbf{x}))^T \cdot \mathbf{d} \leq \varepsilon$;
- The solution is $\mathbf{x}$.

Theorem 4.13. The Newton's method is a **quadratic convergent** method.

4.2 Unconstrained Optimization

Definition 4.14. Given a convex function $f : \mathbb{R}^n \to \mathbb{R}$, a **unconstrained optimization** method solves the problem

$$\min_{\mathbf{x} \in \mathbb{R}^n} f(\mathbf{x})$$

by

(1) **Initialization**: given initial point $\mathbf{x}$ and acceptable error $\varepsilon > 0$, $\delta > 0$;
(2) **Iteration**: calculate the direction $\mathbf{d}$ and step size α, then let $\mathbf{x} = \mathbf{x} + \alpha\mathbf{d}$;
(3) **Stop condition**: if $\|\nabla f(\mathbf{x})\| \le \varepsilon$ or $U_{\mathbb{R}^n}(\mathbf{x}, \delta)$ includes the exact solution, then the current $\mathbf{x}$ is the solution.

Method 4.15. (Gradient descent method) Given the initial point $\mathbf{x}$ and $\varepsilon > 0$,

- The iteration step is:
  (1) Calculate $\mathbf{d} = -\nabla f(\mathbf{x})$ and step size α by a line search method;
  (2) Let $\mathbf{x} = \mathbf{x} + \alpha\mathbf{d}$;
- The stop condition is $\|\nabla f(\mathbf{x})\| \le \varepsilon$;
- The solution is $\mathbf{x}$.

Theorem 4.16. The gradient descent method is a **linear convergent** method.

Method 4.17. (Quasi-Newton method) Given the initial point $\mathbf{x}$, $\varepsilon > 0$ and a matrix $H \in \mathbb{R}^{n\times n}$ (usually the identity matrix),

- The k-th iteration step is:
  (1) Calculate $\mathbf{d}_k = -H_k\nabla f(\mathbf{x}_k)$ and step size α_k by a line search method;
  (2) Let $\mathbf{x}_k = \mathbf{x}_k + \alpha_k\mathbf{d}_k$ and $H_{k+1} = r_k(H_k)$ where the function r_k is a **update** depends on $\mathbf{x}_k$, $\mathbf{x}_{k+1}$, $\nabla f(\mathbf{x}_k)$ and $\nabla f(\mathbf{x}_{k+1})$;
- The stop condition is $\|\nabla f(\mathbf{x}_k)\| \le \varepsilon$;
- The solution is $\mathbf{x}_k$ that satisfies the stop condition.

Definition 4.18. Let $\mathbf{s}_k = \mathbf{x}_{k+1} - \mathbf{x}_k$ and $\mathbf{y}_k = \nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k)$, the **Symmetric Rank-1 update (SR1)** is

$$H_{k+1} = H_k + \frac{(\mathbf{s}_k - H_k\mathbf{y}_k)(\mathbf{s}_k - H_k\mathbf{y}_k)^T}{(\mathbf{s}_k - H_k\mathbf{y}_k)^T\mathbf{y}_k}.$$

The **DFP update** is a rank-2 update defined as

$$H_{k+1} = H_k + \frac{\mathbf{s}_k\mathbf{s}_k^T}{\mathbf{s}_k^T\mathbf{y}_k} - \frac{H_k\mathbf{y}_k\mathbf{y}_k^T H_k}{\mathbf{y}_k^T H_k\mathbf{y}_K}.$$

The **BFGS update** is a rank-2 update defined as

$$H_{k+1} = H_k + \left(1 + \frac{\mathbf{y}_k^T H_k\mathbf{y}_k}{\mathbf{s}_k^T\mathbf{y}_k}\right)\frac{\mathbf{s}_k\mathbf{s}_k^T}{\mathbf{s}_k^T\mathbf{y}_k} - \frac{\mathbf{s}_k\mathbf{y}_k^T H_k + H_k\mathbf{y}_k\mathbf{s}_k^T}{\mathbf{s}_k^T\mathbf{y}_K}.$$

Theorem 4.19. The Quasi-Newton method is a **superlinear convergent** method.

Method 4.20. (Newton's method) Given the initial point $\mathbf{x}$ and $\varepsilon > 0$,

- The iteration step is:
  (1) Calculate $\mathbf{d} = -(\nabla^2 f(\mathbf{x}))^{-1}\nabla f(\mathbf{x})$ and step size α by a line search method;
  (2) Let $\mathbf{x} = \mathbf{x} + \alpha\mathbf{d}$;
- The stop condition is $\|\nabla f(\mathbf{x})\| \le \varepsilon$;
- The solution is $\mathbf{x}$.

Theorem 4.21. The Newton's method is a **quadratic convergent** method.

Chapter 5

Initial Value Problem

Definition 5.1. For $T \geq 0$, $\mathbf{f} : \mathbb{R}^n \times [0,T] \to \mathbb{R}^n$ and $\mathbf{u}_0 \in \mathbb{R}^n$, the **initial value problem** (IVP) is to find $u(t) \in C^1$ satisfies

$$\mathbf{u}' = \mathbf{f}(\mathbf{u}(t), t), \quad \mathbf{u}(0) = \mathbf{u}_0.$$

Notation 5.2. To numerically solve the IVP, we are given initial condition $\mathbf{u}_0 = \mathbf{u}(t_0)$, and want to compute approximations $\{\mathbf{u}_k, k = 1, 2, ...\}$ such that

$$\mathbf{u}_k \approx \mathbf{u}(t_k),$$

where k is the uniform time step size and $t_n = nk$.

5.1 Linear Multistep Method

Definition 5.3. For solving the IVP, an s-step **linear multistep method** (LMM) has the form

$$\sum_{j=0}^{s} \alpha_j \mathbf{u}_{n+j} = k \sum_{j=0}^{s} \beta \mathbf{f}\left(\mathbf{u}_{n+j}, t_{n+j}\right),$$

where $\alpha_s = 1$ is assumed WLOG.

Definition 5.4. An LMM is **explicit** if $\beta_s = 0$, otherwise it is **implicit**.

5.2 Runge-Kutta Method

Definition 5.5. An s-stage **Runge-Kutta method** (RK) is a one-step method of the form

$$\begin{aligned}
\mathbf{y}_i &= \mathbf{f}\left(\mathbf{u}_n + k \sum_{j=1}^{s} a_{ij} \mathbf{y}_j, t_n + c_i k\right), \\
\mathbf{u}_{i+1} &= \mathbf{u}_i + k \sum_{j=1}^{s} b_j \mathbf{y}_j,
\end{aligned}$$

where $i = 1, ..., s$ and $a_{ij}, b_j, c_i \in \mathbb{R}$.

Definition 5.6. The textsf{Butcher tableau} is one way to organize the coefficients of an RK method as follows

$$\begin{array}{c|ccc}
c_1 & a_{11} & \cdots & a_{1s} \\
\vdots & \vdots & & \vdots \\
c_s & a_{s1} & \cdots & a_{ss} \\
\hline
 & b_1 & \cdots & b_s
\end{array}$$

The matrix $A = \left(a_{ij}\right)_{s \times s}$ is called the RK matrix and $\mathbf{b} = \left(b_1, ..., b_s\right)^T, \mathbf{c} = \left(c_1, ..., c_s\right)^T$ are called the RK weights and the RK nodes.

Definition 5.7. An s-stage **collocation method** is a numerical method for solving the IVP, where we

(1) choose s distinct collocation parameters $c_1, ..., c_s$,

(2) seek s-degree polynomial p satisfying
$\forall i = 1, 2, ..., s, \quad \mathbf{p}(t_n) = \mathbf{u}_n$ and $\mathbf{p}'(t_n + c_i k) = \mathbf{f}(\mathbf{p}(t_n + c_i k), t_n + c_i k)$,

(3) set $\mathbf{u}_{n+1} = \mathbf{p}(t_{n+1})$.

Theorem 5.8. The s-stage collocation method is an s-stage IRK method with

$$a_{ij} = \int_0^{c_i} l_j(\tau)\mathrm{d}\tau, \quad b_j = \int_0^1 l_j(\tau)\mathrm{d}\tau,$$

where $i, j = 1, ..., s$ and $l_k(\tau)$ is the elementary Lagrange interpolation polynomial.

5.3 Theoretical analysis

Definition 5.9. A function $\mathbf{f} : \mathbb{R}^n \times [0, +\infty) \to \mathbb{R}^n$ is **Lipschitz continuous** in its first variable over some domain

$$\Omega = \{(\mathbf{u}, t) : \|\mathbf{u} - \mathbf{u}_0\| \leq a, t \in [0, T]\}$$

iff

$$\exists L \geq 0, \text{ s.t. } \forall(\mathbf{u}, t) \in \Omega, \quad \|\mathbf{f}(\mathbf{u}, t) - \mathbf{f}(\mathbf{v}, t) \leq \|\mathbf{u} - \mathbf{v}\|.$$

5.3.1 Error analysis

Definition 5.10. The **local truncation error** τ is the error caused by replacing continuous derivatives with numerical formulas.

Definition 5.11. A numerical formulas is **consistent** if $\lim\limits_{k\to 0} \tau = 0$.

5.3.2 Stability

Definition 5.12. The **region of absolute stability** (RAS) of a numerical method, applied to

$$\mathbf{u}' = \lambda \mathbf{u}, \quad \mathbf{u}_0 = \mathbf{u}(t_0),$$

is the region Ω that

$$\forall \mathbf{u}_0, \quad \forall \lambda k \in \Omega, \quad \lim_{n\to+\infty} \mathbf{u}_n = 0.$$

Definition 5.13. The **stability function** of a one-step method is a function $R : \mathbb{C} \to \mathbb{C}$ that satisfies

$$\mathbf{u}_{n+1} = R(z)\mathbf{u}_n$$

for the $\mathbf{u}' = \lambda \mathbf{u}$ where Re $(E(\lambda)) \leq 0$ and $z = k\lambda$.

Definition 5.14. A numerical method is **stable** or **zero stable** iff its application to any IVP with $\mathbf{f}(\mathbf{u}, t)$ Lipschitz continuous in $\mathbf{u}$ and continuous in t yields

$$\forall T > 0, \quad \lim_{k\to 0, Nk=t} \|\mathbf{u}_n\| < \infty.$$

Definition 5.15. A numerical method is **A($\boldsymbol{\alpha}$)-statble** if the region of absolute stability Ω satisfies

$$\{z \in \mathbb{C} : \pi - \alpha \leq \arg(z) \leq \pi + \alpha\} \subseteq \Omega.$$

Definition 5.16. A numerical method is **A-statble** if the region of absolute stability Ω satisfies

$$\{z \in \mathbb{C} : \text{Re } (z) \leq 0\} \subseteq \Omega.$$

Definition 5.17. A one-step method is **L-stable** if it is A-stable, and its stability function satisfies

$$\lim_{z\to\infty} |R(z)| = 0.$$

Definition 5.18. An one-step method is **I-stable** iff its stability function satisfies

$$\forall y \in \mathbb{R}, |R(y\mathbf{i})| \le 1.$$

Definition 5.19. An one-step method is **B-stable** (or **contractive**) if for any contractive ODE system, every pair of its numerical solutions $\mathbf{u}_n$ and $\mathbf{v}_n$ satisfy

$$\forall n \in \mathbb{N}, \|u_{n+1} - v_{n+1}\| \le \|u_n - v_n\|.$$

Definition 5.20. An RK method is **algebraically stable** iff the RK weights $b_1, ..., b_s$ are nonnegative, the **algebraic stability matrix** $M = \left(b_i a_{ij} + b_i a_{ji} - b_i b_j\right)_{s\times s}$ is positive semidefinite.

Theorem 5.21. The order of accuracy of an implicit A-stable LMM satisfies $p \le 2$. An explicit LMM cannot be A-stable.

Theorem 5.22. No ERK method is A-stable.

Theorem 5.23. An RK method is A-stable if and only if it is I-stable and all poles of its stability function $R(z)$ have positive real parts.

Theorem 5.24. If an A-stable RK method with a nonsingular RK matrix A is stiffly accurate, then it is L-stable.

Theorem 5.25. If an A-stable RK method with a nonsingular RK matrix A satisfies

$$\forall i \in \{1, ..., s\}, \quad a_{i1} = b_i,$$

then it is L-stable.

Theorem 5.26. B-stable one-step methods are A-stable.

Theorem 5.27. An algebraically stable RK method is B-stable and A-stable.

5.3.3 Convergence

Definition 5.28. A numerical method is convergent iff its application to any IVP with $\mathbf{f}(\mathbf{u}, t)$ Lipschitz continuous in $\mathbf{u}$ and continuous in t yields

$$\forall T > 0, \quad \lim_{k\to 0, nk=T} \mathbf{u}_n = \mathbf{u}(T).$$

Theorem 5.29. A numerical method is convergent iff it is consistent and stable.

5.4 Important Methods

5.4.1 Forward Euler's method

Definition 5.30. The **forward Euler's method** solves the IVP by

$$\mathbf{u}_{n+1} = \mathbf{u}_n + k\mathbf{f}(\mathbf{u}_n, t_n).$$

Theorem 5.31. The region of absolute stability for forward Euler's method is

$$\{z \in \mathbb{C} : |1 + z| \leq 1\}.$$

5.4.2 Backward Euler's method

Definition 5.32. The **backward Euler's method** solves the IVP by

$$\mathbf{u}_{n+1} = \mathbf{u}_n + k\mathbf{f}(\mathbf{u}_{n+1}, t_{n+1}).$$

Theorem 5.33. The region of absolute stability for backward Euler's method is

$$\{z \in \mathbb{C} : |1 - z| \geq 1\}.$$

5.4.3 Trapezoidal method

Definition 5.34. The **trapezoidal method** solves the IVP by

$$\mathbf{u}_{n+1} = \mathbf{u}_n + \frac{k}{2}(\mathbf{f}(\mathbf{u}_n, t_n) + \mathbf{f}(\mathbf{u}_{n+1}, t_{n+1})).$$

Theorem 5.35. The region of absolute stability for trapezoidal method is

$$\left\{ z \in \mathbb{C} : \left| \frac{2+z}{2-z} \right| \geq 1 \right\}.$$

5.4.4 Midpoint method (Leapfrog method)

Definition 5.36. The **midpoint method (Leapfrog method)** solves the IVP by

$$\mathbf{u}_{n+1} = \mathbf{u}_{n-1} + 2k\mathbf{f}(\mathbf{u}_n, t_n).$$

Theorem 5.37. The region of absolute stability for midpoint method is

$$\left\{ z \in \mathbb{C} : \left| z \pm \sqrt{1 + z^2} \right| \leq 1 \right\} \stackrel{?}{=} \{0\}.$$

5.4.5 Heun's third-order RK method

Definition 5.38. The **Heun's third-order formula** is an ERK method of the form

$$\begin{cases} \mathbf{y}_1 &= \mathbf{f}(\mathbf{u}_n, t_n), \\ \mathbf{y}_2 &= \mathbf{f}\left(\mathbf{u}_n + \frac{k}{3}\mathbf{y}_1, t_n + \frac{k}{3}\right), \\ \mathbf{y}_3 &= \mathbf{f}\left(\mathbf{u}_n + \frac{2k}{3}\mathbf{y}_2, t_n + \frac{2k}{3}\right), \\ \mathbf{u}_{n+1} &= \mathbf{u}_n + \frac{k}{4}(\mathbf{y}_1 + 3\mathbf{y}_3). \end{cases} \qquad \begin{array}{c|ccc} 0 & 0 & 0 & 0 \\ \frac{1}{3} & \frac{1}{3} & 0 & 0 \\ \frac{2}{3} & 0 & \frac{2}{3} & 0 \\ \hline & \frac{1}{4} & 0 & \frac{3}{4} \end{array}$$

5.4.6 Classical fourth-order RK method

Definition 5.39. The **classical fourth-order RK method** is an ERK method of the form

$$\begin{cases} \mathbf{y}_1 &= \mathbf{f}(\mathbf{u}_n, t_n), \\ \mathbf{y}_2 &= \mathbf{f}\Big(\mathbf{u}_n + \frac{k}{2}\mathbf{y}_1, t_n + \frac{k}{2}\Big), \\ \mathbf{y}_3 &= \mathbf{f}\Big(\mathbf{u}_n + \frac{k}{2}\mathbf{y}_2, t_n + \frac{k}{2}\Big), \\ \mathbf{y}_4 &= \mathbf{f}(\mathbf{u}_n + k\mathbf{y}_3, t_n + k), \\ \mathbf{u}_{n+1} &= \mathbf{u}_n + \frac{k}{6}(\mathbf{y}_1 + 2\mathbf{y}_2 + 2\mathbf{y}_3 + \mathbf{y}_4). \end{cases} \qquad \begin{array}{c|cccc} 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array}$$

5.4.7 TR-BDF2 method

Definition 5.40. The **TR-BDF2 method** is an one-step method of the form

$$\begin{cases} \mathbf{u}_* &= \mathbf{u}_n + \frac{k}{4}\Big(\mathbf{f}(\mathbf{u}_n, t_n) + \mathbf{f}\Big(\mathbf{u}_*, t_n + \frac{k}{2}\Big)\Big), \\ \mathbf{u}_{n+1} &= \frac{1}{3}(4\mathbf{u}_* - \mathbf{u}_n + k\mathbf{f}(\mathbf{u}_{n+1}, t_{n+1})). \end{cases}$$

Chapter 6

Number Theory

6.1 Prime Number

Definition 6.1. A **prime number** (or a **prime**) is a natural number greater than 1 that is not a product of two smaller natural numbers.

Definition 6.2. A **composite number** (or a **composite**) is a natural number greater than 1 that is a product of two smaller natural numbers.

6.1.1 Primality testing

Theorem 6.3. For a integer $n \in \mathbb{N}$, if it is a product of two natural number a and b thar $a \leq b$, then

$$1 \leq a \leq \sqrt{n} \leq b \leq n.$$

Method 6.4. (Trial division) Given a integer n, the **trial division method** divides n by each integer from 2 up to $\sqrt{n}$. Any such integer dividing n evenly establishes n as composite, otherwise it is prime.

Theorem 6.5. (Fermat's little theorem) For a prime number p and a number a that gcd $(a, p) = 1$, then $a^{p-1} \equiv 1 (\text{mod } p)$

Method 6.6. The **Miller-Rabin** algorithm is a method of primality testing, where given a number n, where we

(1) determine directly for small numbers such as $p = 2$.
(2) factorize the number $p = u \times 2^t$;
(3) choose a number a that gcd $(a, p) = 1$, and calculate $a^u, a^{u\times 2}, a^{u\times 2^2}, ..., a^{u\times 2^{t-1}}$;
(4) if $a^u \equiv 1 (\text{mod } p)$, or $\exists a^{u\times k}, k < t$ that $a^{u\times k} \equiv p - 1 (\text{mod } p)$ then p passes the test, otherwise, p is a composite number;
(5) repeat above steps to eliminate error.

For numbers less than 2^{32}, choose $a \in \{2, 7, 61\}$ is enough, for numbers less than $2^{\{64\}}$, choose $a \in \{2, 325, 9375, 28178, 450775, 9780504, 1795265022\}$ is enough.

6.1.2 Sieves

Method 6.7. (Sieve of Eratosthenes) Given a upper limit n, the **sieve of Eratosthenes** solves all the prime numbers up to n by marking composite numbers, where we

(1) create a list of consecutive integers from 2 to n: $\{2, 3, 4, ..., n\}$;
(2) initially, let $p = 2$, the smallest prime number;
(3) enumerate the multiples of p by counting in increments of p from $2p$ to n, and mark them in the list;
(4) find the smallest number in the list greater than p that is not marked;
(5) if there was no such number, the method is terminated and the numbers remaining not marked in the list are all the primes below n, otherwise let p now equal the new number which is the next prime, and repeat from step (3).