# Introduction to numerical analysis

LIM, Roktaek
Ulsan National Institute of Science and Technology

2024 Summer School on Numerical Relativity and Gravitational Waves

# Part 1. Preliminaries

# What is numerical analysis?

The study of **algorithms** for the problems of continuous mathematics.

(Lloyd N. Trefethen, *SIAM News*, 1992)

# What is numerical analysis?

The study of **algorithms** for the problems of continuous mathematics.

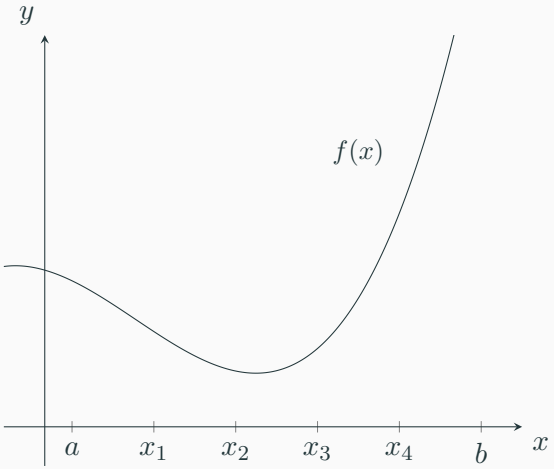
(Lloyd N. Trefethen, *SIAM News*, 1992)

The field concerned with the design of computable **algorithms** for solving mathematical problems, and with the analysis of their accuracy, efficiency, and other aspects of performance.

(D. Arnold, 2024 Simons Conference on Localization of Waves)

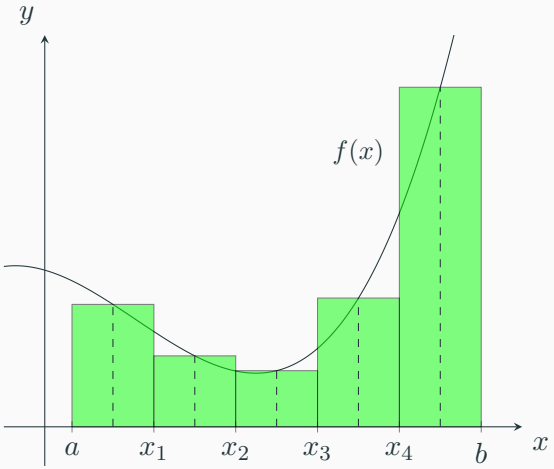
# An example: Computation of a definite integral

Let $a$ and $b$ be two real numbers with $a < b$. A continuous function $f : [a, b] \to \mathbb{R}$ is given. How to compute $\int_a^b f(x)\ dx$?



# An example: Computation of a definite integral

Let $a$ and $b$ be two real numbers with $a < b$. A continuous function $f : [a, b] \to \mathbb{R}$ is given. How to compute $\int_a^b f(x)\, dx$?



# An example: Computation of a definite integral

Let $a$ and $b$ be two real numbers with $a < b$. A continuous function $f : [a, b] \to \mathbb{R}$ is given. How to compute $\int_a^b f(x)\, dx$?

$$\int_a^b f(x)\, dx \approx Q(f) = \sum_{j=1}^{N} hf\left(\bar{x}_j\right)$$

where $h = (b - a)/N$ and $\bar{x}_j = a + (j - 0.5)h$ for $j = 1, \ldots, N$.

---

The composite midpoint rule

---

$h = (b - a)/N$, $x = a - 0.5h$, and $Q = 0$.
**for** $j = 1$ to $N$ **do**
  $x = x + h$
  $Q = Q + hf(x)$
**end for**

---

# An example: Computation of a definite integral

Let $a$ and $b$ be two real numbers with $a < b$. A continuous function $f : [a, b] \to \mathbb{R}$ is given. How to compute $\int_a^b f(x)\ dx$?

$$\int_a^b f(x)\ dx \approx Q(f) = \sum_{j=1}^{N} hf\left(\bar{x}_j\right)$$

where $h = (b - a)/N$ and $\bar{x}_j = a + (j - 0.5)h$ for $j = 1, \ldots, N$.

The definite integral of a continuous function $f(x)$ over the interval $[a, b]$ is the limit of a Riemann sum as the number of subdivisions approaches infinity,

$$\int_a^b f(x)\ dx = \lim_{N \to \infty} \sum_{j=1}^{N} hf\left(\bar{x}_j\right).$$

# Computable algorithms

$$\int_a^b f(x)\,dx \approx Q(f) = \sum_{j=1}^{N} hf\left(\bar{x}_j\right), \tag{$\star$}$$

$$\int_a^b f(x)\,dx = \lim_{N\to\infty} \sum_{j=1}^{N} hf\left(\bar{x}_j\right). \tag{$\dagger$}$$

($\star$) can be implemented on computers, but ($\dagger$) cannot.

# Computable algorithms

$$\int_a^b f(x)\,dx \approx Q(f) = \sum_{j=1}^{N} hf\left(\bar{x}_j\right), \tag{$\star$}$$

$$\int_a^b f(x)\,dx = \lim_{N\to\infty} \sum_{j=1}^{N} hf\left(\bar{x}_j\right). \tag{$\dagger$}$$

($\star$) can be implemented on computers, but ($\dagger$) cannot.

Computers can perform only a **finite** sequence of operations.

# Can we exactly compute $Q(f)$ on computers?

$$Q(f) = \sum_{j=1}^{N} hf\left(\bar{x}_j\right) \qquad (\star)$$

where $h = (b-a)/N$ and $\bar{x}_j = a + (j-0.5)h$ for $j = 1, \ldots, N$.

# Can we exactly compute $Q(f)$ on computers?

$$Q(f) = \sum_{j=1}^{N} h f\left(\bar{x}_j\right) \qquad (\star)$$

where $h = (b-a)/N$ and $\bar{x}_j = a + (j-0.5)h$ for $j = 1, \ldots, N$.

- Real numbers cannot be represented exactly on computer.
  $\rightarrow$ We discretize real numbers by introducing of floating numbers.

# Machine epsilon

**Machine epsilon test using Python**

```
>>> import sys
>>> eps_m = sys.float_info.epsilon
>>> 1.0 + 0.5*eps_m
1.0
>>> 1.0 - 0.5*eps_m
0.9999999999999999
>>> (1.0 + 2/eps_m) - 2/eps_m
0.0
```

# Floating-point representation

- A computer must store a finite amount of data.
- All numbers and arithmetic are done with some error.
- Sometime, this finite precision issue is minor.
- However, it is important to be concerned with the effect of finite precision arithmetic on numerical algorithms.

# Floating-point representation

The floating numbers can be represented as

$$x = (-1)^s \times 2^e \times m.$$

This notation has three parts:

- a sign $s$,
- a mantissa in the interval $[1, 2)$, $m$,
- an exponent $e$.

For example,

$$\begin{aligned} -4512 &= (-1)^1 \times \left(2^{12} + 2^8 + 2^7 + 2^5\right) \\ &= (-1)^1 \times 2^{12} \times \left(1 + 2^{-4} + 2^{-5} + 2^{-7}\right) \\ &= (-1)^1 \times 2^{12} \times (1.0001101)_2 \end{aligned}$$

# Floating-point representation

Some references for the floating-point representation:

- Chap 2 and 9 in 754-2019 - IEEE Standard for Floating-Point Arithmetic (`https://ieeexplore.ieee.org/document/8766229`),
- Chap 1 and 2 in Accuracy and stability of numerical algorithms (`https://epubs.siam.org/doi/book/10.1137/1.9780898718027`),
- D. Goldberg, What every computer scientist should know about floating-point arithmetic (`https://doi.org/10.1145/103162.103163`).

**Absolute and relative errors**

Let $\hat{x}$ be an approximation to a real number $x$. Then its absolute error is given by

$$E_{\text{abs}}(\hat{x}) = |x - \hat{x}|$$

and its relative error is defined as

$$E_{\text{rel}}(\hat{x}) = \frac{|x - \hat{x}|}{|x|}.$$

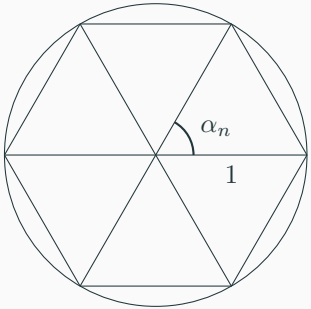
# Roundoff error

Suppose that a positive real number $x$ does not have an exact representation as a floating point number:

$$x_- = 1.b_1 b_2 \dots b_{n-1} b_n \times 2^m < x < x_- + 0.00 \dots 01 \times 2^m = x_+.$$

The process of replacing the real number $x$ by a nearby machine number (either $x_-$ or $x_+$) is called rounding, and the error involved is called **roundoff error**.

# Roundoff error: Examples




## Area of $n$-gon

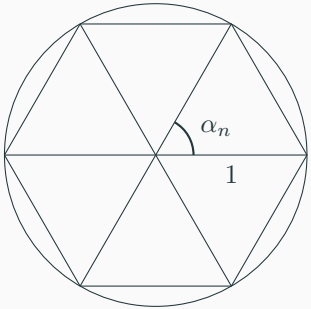
$$A_n = n \cos\frac{\alpha_n}{2} \sin\frac{\alpha_n}{2} = \frac{n}{2}\sin\alpha_n = \frac{n}{2}\sin\left(\frac{2\pi}{n}\right).$$

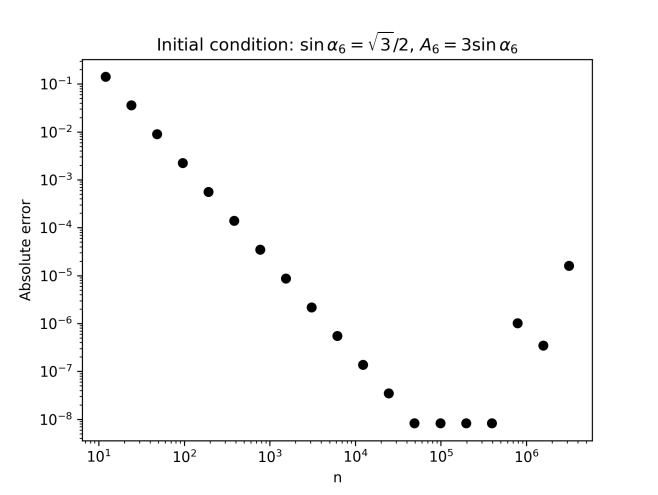
$$A_{2n} = \frac{n}{2}\sin\frac{\alpha_n}{2}$$

where

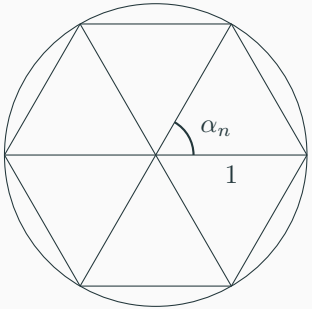
$$\sin\frac{\alpha_n}{2} = \sqrt{\frac{1-\cos\alpha_n}{2}} = \sqrt{\frac{1-\sqrt{1-\sin^2\alpha_n}}{2}}.$$

# Roundoff error: Examples



# Roundoff error: Examples



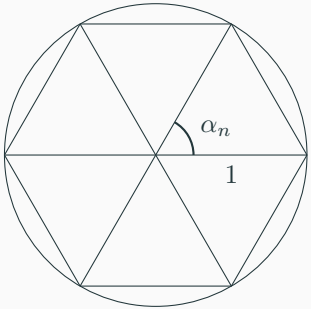

## $\sin \frac{\alpha_n}{2}$

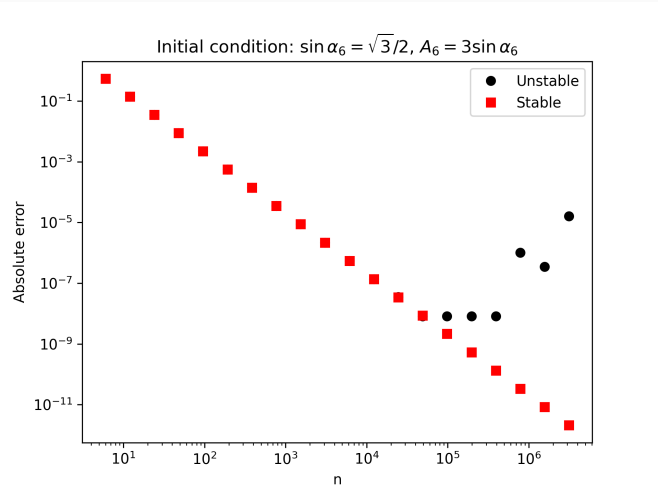
$$\sin \frac{\alpha_n}{2} = \sqrt{\frac{1 - \sqrt{1 - \sin^2 \alpha_n}}{2}}.$$

For $\alpha_n \ll 1$, $\sqrt{1 - \sin^2 \alpha_n} \approx 1$.

$$\begin{aligned}\sin \frac{\alpha_n}{2} &= \sqrt{\frac{1 - \sqrt{1 - \sin^2 \alpha_n}}{2} \left( \frac{1 + \sqrt{1 - \sin^2 \alpha_n}}{1 + \sqrt{1 - \sin^2 \alpha_n}} \right)} \\ &= \frac{\sin \alpha_n}{\sqrt{2 \left( 1 + \sqrt{1 - \sin^2 \alpha_n} \right)}}\end{aligned}$$

# Roundoff error: Examples



# Can we exactly compute $Q(f)$ on computers?

$$Q(f) = \sum_{j=1}^{N} h f\left(\bar{x}_j\right) \qquad (\star)$$

where $h = (b-a)/N$ and $\bar{x}_j = a + (j - 0.5)h$ for $j = 1, \ldots, N$.

- Real numbers cannot be represented exactly on computer.
  $\rightarrow$ We discretize real numbers by introducing of floating numbers.
- Computers do not know functions such as $\sin x$.
  $\rightarrow$ We discretize functions by interpolation or curve fitting.

# CORDIC (coordinate rotation digital computer)

Meher *et al*. 50 years of CORDIC: Algorithms, architectures, and applications, IEEE Transactions on Circuits and Systems I: Regular Papers 56.9 (2009): 1893-1907. `https://doi.org/10.1109/TCSI.2009.2025803`

**Abstract**

Year 2009 marks the completion of 50 years of the invention of CORDIC (coordinate rotation digital computer) by Jack E. Volder. The beauty of CORDIC lies in the fact that by simple shift-add operations, it can perform several computing tasks such as the calculation of trigonometric, hyperbolic and logarithmic functions, ...

# The concept of condition

A problem can be viewed as a mapping $F : X \to Y$. Here $X$ is a data space of a given problem and $Y$ is a result space of a given problem.

- Solving a linear systems of equations $A\mathbf{x} = \mathbf{b}$.
  $\to X = \{A, \mathbf{b}\}$ and $Y = \mathbf{x}$.
- Computing the roots of a polynomial of order $n$ with real coefficients.
  $\to X$ consists of polynomial coefficients $c_0, c_1, \ldots, c_n$ and $Y$ consists of the $n$ roots of the polynomial.

# The concept of condition

**Condition number**

Condition number is a measure of sensitivity of a problem. Suppose that the input value $x \in X$ is changed by $\delta x \leq \epsilon$. Then, the output $F(x)$ changes by $F(x + \delta x) - F(x)$. The relative condition number is defined as

$$\kappa = \lim_{\epsilon \to 0} \sup_{|\delta x| \epsilon \leq \epsilon} \left( \frac{|(F(x + \delta x) - F(x))\,/F(x)|}{|\delta x / x|} \right).$$

A problem is called **well-conditioned** if small changes in the input lead to small changes in the output.
If a small error made in the input can lead to a drastic difference in the output, the problem is **ill-conditioned**.

**Ill-conditioned problem**

Consider the problem of evaluating

$$f(x) = \tan x.$$

Take $x_1 = \frac{\pi}{2} - 0.001$ and $x_1 = \frac{\pi}{2} - 0.002$.
$|x_1 - x_2| = 0.001$ and $|f(x_1) - f(x_2)| \approx 500.0$. Then, $\kappa \approx 500.0$. The small difference in $x$ leads to large differences in $f$. This problem is ill-conditioned.

The condition number is a property of the examined **problem**, not of the used algorithm.

# The concept of stability

An algorithm can be viewed as another mapping $\hat{F} : X \to Y$. If $\hat{F}$ for a given $F$ is accurate for each $x \in X$,

$$\frac{|F(x) - \hat{F}(x)|}{|F(x)|} \leq \epsilon.$$

**Stability**

An algorithm $\hat{F}$ for a problem $F$ is stable if for each $x \in X$,

$$\frac{|F(\hat{x}) - \hat{F}(x)|}{|F(\hat{x})|} \leq \epsilon$$

for some $\hat{x}$ with

$$\frac{|\hat{x} - x|}{|x|} \leq \epsilon.$$

# The concept of stability

- Input data can be perturbed. However, they are in a neighborhood of exact input $x$, $|\hat{x} - x|/|x| \leq \epsilon$.
- Thus, any such $\hat{x}$ has to be considered as virtually equal to $x$.
- An algorithm is said to be **stable** if small errors in the inputs and at each step lead to small errors in the solution.
- If an algorithm is stable, $\hat{F}(x)$ is in a neighborhood of $F(\hat{x})$.
- An algorithm that amplifies errors is called **unstable**.

# The concept of stability

$$e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!} = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \cdots$$

Approximation of $e^x$

**Input:** $x$ and $\mathrm{tol}$

$y = 1, p = 1$, and $k = 1$

**while** $|p| > \mathrm{tol} * y$ **do**

$p = p * x/k$

$y = y + p$

$k = k + 1$

**end while**

| | x | y_exact | y_approx | rel_error |
|---|---|---|---|---|
| 0 | -20.0 | 2.061154e-09 | 6.147562e-09 | 1.982583e+00 |
| 1 | -18.0 | 1.522998e-08 | 1.598372e-08 | 4.949059e-02 |
| 2 | -16.0 | 1.125352e-07 | 1.124750e-07 | 5.344260e-04 |
| 3 | -14.0 | 8.315287e-07 | 8.315442e-07 | 1.859183e-05 |
| 4 | -12.0 | 6.144212e-06 | 6.144211e-06 | 2.993215e-07 |
| 5 | -10.0 | 4.539993e-05 | 4.539993e-05 | 3.501044e-09 |
| 6 | -8.0 | 3.354626e-04 | 3.354626e-04 | 6.620042e-10 |
| 7 | -6.0 | 2.478752e-03 | 2.478752e-03 | 3.325188e-10 |
| 8 | -4.0 | 1.831564e-02 | 1.831564e-02 | 5.307235e-10 |
| 9 | -2.0 | 1.353353e-01 | 1.353353e-01 | 2.736031e-10 |
| 10 | 0.0 | 1.000000e+00 | 1.000000e+00 | 0.000000e+00 |
| 11 | 2.0 | 7.389056e+00 | 7.389056e+00 | 4.799685e-10 |
| 12 | 4.0 | 5.459815e+01 | 5.459815e+01 | 1.923058e-09 |
| 13 | 6.0 | 4.034288e+02 | 4.034288e+02 | 1.344248e-09 |
| 14 | 8.0 | 2.980958e+03 | 2.980958e+03 | 2.102584e-09 |
| 15 | 10.0 | 2.202647e+04 | 2.202647e+04 | 2.143799e-09 |
| 16 | 12.0 | 1.627548e+05 | 1.627548e+05 | 1.723845e-09 |
| 17 | 14.0 | 1.202604e+06 | 1.202604e+06 | 3.634135e-09 |
| 18 | 16.0 | 8.886111e+06 | 8.886111e+06 | 2.197990e-09 |
| 19 | 18.0 | 6.565997e+07 | 6.565997e+07 | 3.450972e-09 |
| 20 | 20.0 | 4.851652e+08 | 4.851652e+08 | 4.828737e-09 |

# The concept of stability

$$e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!} = 1 + x + \frac{x^2}{2} + \frac{x^3}{6} + \cdots$$

Approximation of $e^x$

**Input:** $x$ and $\mathrm{tol}$

$y = 1, p = 1$, and $k = 1$

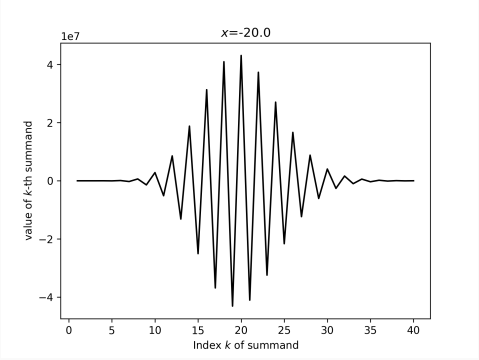
**while** $|p| > \mathrm{tol} * y$ **do**

$p = p * x/k$

$y = y + p$

$k = k + 1$

**end while**



# The concept of stability

We can have a stable algorithm by using the identity $e^x = 1/e^{-x}$ when $x < 0$.

---

Approximation of $e^x$ when $x < 0$

---

**Input:** $x$ and $\mathrm{tol}$

$y = 1, p = 1$, and $k = 1$

**while** $|p| > \mathrm{tol} * y$ **do**

$p = p * (-x)/k$

$y = y + p$

$k = k + 1$

**end while**

$y = 1/y$

---

| | x | y_exact | y_approx | rel_error |
|---|---|---|---|---|
| 0 | -20.0 | 2.061154e-09 | 2.061154e-09 | 4.828737e-09 |
| 1 | -18.0 | 1.522998e-08 | 1.522998e-08 | 3.450972e-09 |
| 2 | -16.0 | 1.125352e-07 | 1.125352e-07 | 2.197990e-09 |
| 3 | -14.0 | 8.315287e-07 | 8.315287e-07 | 3.634135e-09 |
| 4 | -12.0 | 6.144212e-06 | 6.144212e-06 | 1.723845e-09 |
| 5 | -10.0 | 4.539993e-05 | 4.539993e-05 | 2.143799e-09 |
| 6 | -8.0 | 3.354626e-04 | 3.354626e-04 | 2.102584e-09 |
| 7 | -6.0 | 2.478752e-03 | 2.478752e-03 | 1.344248e-09 |
| 8 | -4.0 | 1.831564e-02 | 1.831564e-02 | 1.923058e-09 |
| 9 | -2.0 | 1.353353e-01 | 1.353353e-01 | 4.799683e-10 |
| 10 | 0.0 | 1.000000e+00 | 1.000000e+00 | 0.000000e+00 |
| 11 | 2.0 | 7.389056e+00 | 7.389056e+00 | 4.799685e-10 |
| 12 | 4.0 | 5.459815e+01 | 5.459815e+01 | 1.923058e-09 |
| 13 | 6.0 | 4.034288e+02 | 4.034288e+02 | 1.344248e-09 |
| 14 | 8.0 | 2.980958e+03 | 2.980958e+03 | 2.102584e-09 |
| 15 | 10.0 | 2.202647e+04 | 2.202647e+04 | 2.143799e-09 |
| 16 | 12.0 | 1.627548e+05 | 1.627548e+05 | 1.723845e-09 |
| 17 | 14.0 | 1.202604e+06 | 1.202604e+06 | 3.634135e-09 |
| 18 | 16.0 | 8.886111e+06 | 8.886111e+06 | 2.197990e-09 |
| 19 | 18.0 | 6.565997e+07 | 6.565997e+07 | 3.450972e-09 |
| 20 | 20.0 | 4.851652e+08 | 4.851652e+08 | 4.828737e-09 |

# Part 2. Numerical quadrature

# Numerical quadrature

Let $f(x)$ be a real-valued function of a real variable, defined on a finite interval $[a, b]$. Our aim is to compute the value of the integral

$$\int_a^b f(x)\, dx.$$

- Quadrature is a historical term which means determining area.
- Numerical quadrature is computing the approximate evaluation of such a definite integral.

# Numerical quadrature

Almost all rules of quadratures can be written as weighted sums of function values

$$\int_a^b f(x)\,dx \approx Q(f) = \sum_{j=0}^{n} w_j f(x_j)$$

with weights $w_j$ and pairwise different nodes $x_j$, where

$$a = x_0 < x_1 < \cdots < x_{n-1} < x_n = b.$$

# Numerical quadrature

- Numerical quadrature is based on polynomial interpolation.
- Integrand function $f$ is sampled at a finite set of points, $x_0, x_1, \ldots, x_n$.
- Interpolating polynomial $p(x)$ of $f(x)$ for the nodes $x_j$ is chosen.
- In practice, interpolating polynomial is not determined explicitly but used to determine weights corresponding to nodes.

# Interpolation

Suppose that an unknown function $f(x)$ with values at $n$ distinct points $x_0 < x_1 < x_2 < \cdots < x_n$. This means $f(x_0), \ldots, f(x_n)$ are given. The interpolation problem is to construct a function $I(x)$ that passes through these points.

$$I(x_j) = f(x_j),\ 0 \leq j \leq n. \tag{1}$$

- The points $x_0, x_1, x_2, \ldots, x_n$ are called the interpolation points.
- The property of passing through these points is referred to as interpolating the data.
- The function that interpolates the data is an interpolant or an interpolating polynomial.

# Lagrange form of the interpolation polynomial

Let

$$I_n(x) = \sum_{j=0}^{n} f(x_j) L_j^n(x)$$

where $L_j^n(x)$ are $n+1$ polynomials of degree $n$. By (1),

$$L_j^n(x_i) = \delta_{i,j}, \ i, j = 0, \ldots, n. \tag{2}$$

An obvious way of constructing polynomials $L_j^n(x)$ of degree $n$ that satisfy is the following:

$$L_j^n(x) = \frac{(x - x_0) \cdots (x - x_{j-1})(x - x_{j+1}) \cdots (x - x_n)}{(x_j - x_0) \cdots (x_j - x_{j-1})(x_j - x_{j+1}) \cdots (x_j - x_n)}, \ 0 \leq j \leq n. \tag{3}$$

# Lagrange polynomial

If the Lagrange polynomials $L_j^n(x)$ is used for the representation of $p(x)$, then

$$Q(f(x)) = \int_a^b p(x)\ dx = \int_a^b \sum_{j=0}^{n} f(x_j) L_j^n(x)\ dx$$
$$= \sum_{j=0}^{n} \left( f(x_j) \int_a^b L_j^n(x)\ dx \right).$$

The weights are determined by

$$w_j = \int_a^b L_j(x)\ dx,\ j = 0, \ldots, n.$$

# Lagrange polynomial

Since the interpolation polynomial is unique,

$$\sum_{j=0}^{n} L_j^n(x) = 1.$$

Thus,

$$\sum_{j=0}^{n} w_j = \int_a^b \sum_{j=0}^{n} L_j^n(x)\, dx = b - a.$$

This leads to the sum of the weights is always equal to $b - a$ when the interpolating polynomial is used for numerical quadrature.

# Condition of numerical quadrature

Suppose all input fluctuations are less than $\epsilon > 0$,

$$|\delta Q(f)| = \left| \sum_{j=0}^{n} w_j \delta f_j \right| \leq \epsilon \sum_{j=0}^{n} |w_j|$$

where $f_j = f(x_j)$.

- Recall $\sum_{j=0}^{n} w_j = b - a$.
- If $w_j$ are positive for all $j = 0, \ldots, n$, then the order of magnitude of the RHS is $\mathcal{O}(\epsilon)$. The numerical quadrature problem is well-conditioned.
- If some $w_j$ are negative, then the RHS might be very large. The numerical quadrature problem can be ill-conditioned.

# Numerical quadrature

Consider Lagrange interpolation in equidistant nodes in the integration interval $[a, b]$:

$$x_j = a + jh,\ h = \frac{b-a}{n},\ j = 0, \ldots, n.$$

Lagrange polynomials are given by

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

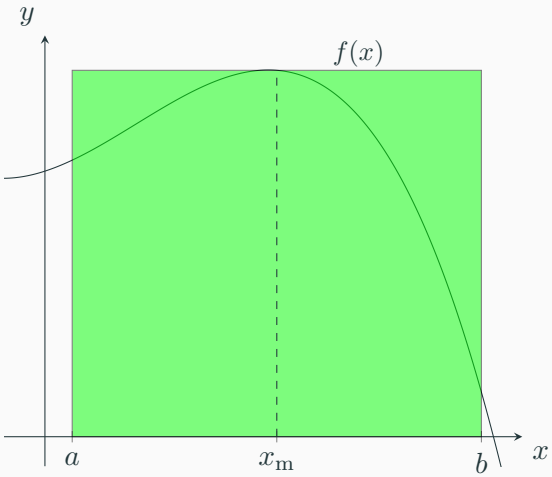
Lagrange interpolating polynomial is given by

$$p_n(x) = \sum_{j=0}^{n} L_j(x) f(x_j).$$

Then,

$$\int_a^b f(x)\ dx \approx \int_a^b p_n(x)\ dx = \sum_{j=0}^{n} f(x_j) \int_a^b L_j(x)\ dx$$

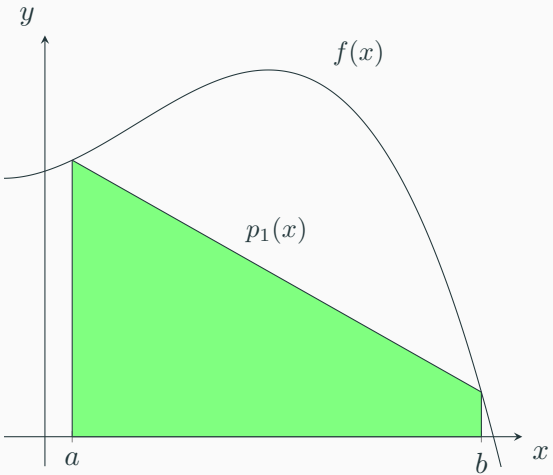
# Midpoint rule




The midpoint rule is for $n = 0$ and $x_{\mathrm{m}} = (b + a)/2$. This leads to the 1-point quadrature formula

$$\int_a^b f(x)\, dx \approx (b - a) f\left(\frac{a + b}{2}\right).$$

# Trapezoidal rule




The trapezoidal rule is for $n = 1$:

$$p_1 = f(a)\frac{x-b}{a-b} + f(b)\frac{x-a}{b-a}$$

with $x_j = a + jh$ and $h = b - a$.

$$\int_a^b \frac{b-x}{b-a}\,dx = \int_a^b \frac{x-a}{b-a}\,dx = \frac{b-a}{2}.$$

Thus,

$$\int_a^b f(x)\,dx \approx \frac{(b-a)}{2}\left(f(a) + f(b)\right).$$

# Simpson's rule

Simpson's rule is for $n = 2$:

$$p_2 = f(a)\frac{(x-b)(x-c)}{(a-b)(a-c)} + f(c)\frac{(x-a)(x-b)}{(c-a)(c-b)} + f(b)\frac{(x-a)(x-c)}{(b-a)(b-c)}$$

with $x_j = a + jh$, $h = b - a$, and $c = (b + a)/2$. Simpson's rule is given by

$$\int_a^b f(x)\, dx \approx \frac{(b-a)}{6}\left(f(a) + 4f(c) + f(b)\right).$$

# Midpoint rule

Suppose $f(x)$ is a smooth function. The Taylor expansion of $f$ at $x_\mathrm{m} = (a+b)/2$ gives

$$f(x) = f(x_\mathrm{m}) + (x - x_\mathrm{m}) f'(x_\mathrm{m}) + \frac{(x - x_\mathrm{m})^2}{2} f''(\xi)$$

where $\xi \in (x_\mathrm{m}, x)$. Integrating the Taylor expansion gives

$$\begin{aligned}\int_a^b f(x)\,dx &= \int_a^b \left( f(x_\mathrm{m}) + (x - x_\mathrm{m}) f'(x_\mathrm{m}) + \frac{(x - x_\mathrm{m})^2}{2} f''(\xi) \right) dx \\ &= (b-a) f(x_\mathrm{m}) + \frac{1}{2} \int_a^b (x - x_\mathrm{m})^2 f''(\xi)\,dx.\end{aligned}$$

This leads to an expression for the error

$$\left| \int_a^b f(x)\,dx - (b-a) f(x_\mathrm{m}) \right| = \frac{1}{2} \left| \int_a^b (x - x_\mathrm{m})^2 f''(\xi)\,dx \right|.$$

# Midpoint rule

$$\left| \int_a^b f(x)\, dx - (b-a) f(x_{\mathrm{m}}) \right| = \frac{1}{2} \left| \int_a^b (x - x_{\mathrm{m}})^2 f''(\xi)\, dx \right|.$$

The RHS is given by

$$\begin{aligned} \frac{1}{2} \left| \int_a^b (x - x_{\mathrm{m}})^2 f''(\xi)\, dx \right| &\leq \frac{1}{2} \int_a^b (x - x_{\mathrm{m}})^2 \left| f''(\xi) \right| \, dx \\ &\leq \frac{M}{2} \int_a^b (x - x_{\mathrm{m}})^2 \, dx \\ &= \frac{M}{24} (b-a)^3 \end{aligned}$$

where $M = \max_{x \in [a,b]} |f''(x)|$.

# Composite midpoint rule

Let $[a, b]$ be partitioned into $n$ equidistant subintervals $(x_j, x_{j+1})$ of length $h = x_{j+1} - x_j = (b - a)/n$. $x_0 = a$ and $x_n = b$.

$$\begin{aligned}\int_a^b f(x)\, dx &= \sum_{j=0}^{n-1} \int_{x_j}^{x_{j+1}} f(x)\, dx \\ &\approx \sum_{j=0}^{n-1} h f\left(\frac{x_j + x_{j+1}}{2}\right).\end{aligned}$$

For each interval $[x_j, x_{j+1}]$,

$$\left| \int_{x_j}^{x_{j+1}} f(x)\, dx - h f\left(\frac{x_j + x_{j+1}}{2}\right) \right| \leq \frac{M_j}{24} h^3$$

where $M_j = \max_{x \in [x_j, x_{j+1}]} |f''(x)|$.

# Composite midpoint rule

Let $M = \max_{0 \le j \le n-1} \{M_j\}$.

$$\left| \int_a^b f(x)\, dx - Q_{\mathsf{mid}}(f) \right| \le \sum_{j=0}^{n-1} \frac{h^3}{24} M.$$

Recall that $h = (b-a)/n$ and $hn = b - a$.

$$\left| \int_a^b f(x)\, dx - Q_{\mathsf{mid}}(f) \right| \le (b-a) \frac{h^2}{24} M.$$

# Trapezoidal rule

The local error in the trapezoidal rule is given by

$$\left| \int_a^b f(x)\, dx - \frac{(b-a)}{2} \left( f(a) + f(b) \right) \right|.$$

Suppose $f$ is a smooth function. The Taylor expansion of $f$ at $x_{\mathrm{m}} = (a+b)/2$ gives

$$\begin{aligned}
f(x) &= f(x_{\mathrm{m}}) + (x - x_{\mathrm{m}}) f'(x_{\mathrm{m}}) + \frac{(x - x_{\mathrm{m}})^2}{2} f''(\xi_1), \\
f(a) &= f(x_{\mathrm{m}}) + (a - x_{\mathrm{m}}) f'(x_{\mathrm{m}}) + \frac{(a - x_{\mathrm{m}})^2}{2} f''(\xi_2), \\
f(b) &= f(x_{\mathrm{m}}) + (b - x_{\mathrm{m}}) f'(x_{\mathrm{m}}) + \frac{(b - x_{\mathrm{m}})^2}{2} f''(\xi_3)
\end{aligned}$$

where $\xi_1 \in (x_{\mathrm{m}}, x)$, $\xi_2 \in (a, x_{\mathrm{m}})$, and $\xi_3 \in (x_{\mathrm{m}}, b)$.

# Trapezoidal rule

Integrating the Taylor expansion gives

$$\int_a^b f(x)\,dx = (b-a)f(x_{\mathrm{m}}) + \frac{1}{2}\int_a^b (x-x_{\mathrm{m}})^2 f''(\xi)\,dx.$$

Note that $a - x_{\mathrm{m}} = -(b-a)/2$ and $b - x_{\mathrm{m}} = (b-a)/2$. Thus,

$$f(a) + f(b) = 2f(x_{\mathrm{m}}) + \frac{(b-a)^2}{8} f''(\xi_2) + \frac{(b-a)^2}{8} f''(\xi_3).$$

We can obtain

$$\left| \int_a^b f(x)\,dx - \frac{(b-a)}{2}\left(f(a)+f(b)\right) \right| \leq \frac{1}{2}\int_a^b (x-x_{\mathrm{m}})^2 \left|f''(\xi_1)\right|\,dx + \frac{(b-a)^3}{16}\left|f''(\xi_2)\right| + \frac{(b-a)^3}{16}\left|f''(\xi_3)\right|.$$

# Trapezoidal rule

$$\left| \int_a^b f(x)\, dx - \frac{(b-a)}{2}\left(f(a)+f(b)\right) \right| \leq \frac{1}{2}\int_a^b (x-x_{\mathrm{m}})^2 \left|f''(\xi_1)\right|\, dx + \frac{(b-a)^3}{16}\left|f''(\xi_2)\right| + \frac{(b-a)^3}{16}\left|f''(\xi_3)\right|.$$

Let $M = \max_{x\in[a,b]} |f''(x)|$. Then,

$$\left| \int_a^b f(x)\, dx - \frac{(b-a)}{2}\left(f(a)+f(b)\right) \right| \leq \frac{M}{6}(b-a)^3.$$

# Composite trapezoidal rule

Let $[a, b]$ be partitioned into $n$ equidistant subintervals $(x_j, x_{j+1})$ of length $h = x_{j+1} - x_j = (b-a)/n$. $x_0 = a$ and $x_n = b$.

$$\begin{aligned}\int_a^b f(x)\,dx &= \sum_{j=0}^{n-1} \int_{x_j}^{x_{j+1}} f(x)\,dx \\ &\approx \sum_{j=0}^{n-1} \frac{h}{2}\left(f(x_j) + f(x_{j+1})\right).\end{aligned}$$

The global error is given by

$$\left| \int_a^b f(x)\,dx - Q_{\text{trap}}(f) \right| \leq (b-a)\frac{h^2}{6} M$$

where $M_j = \max_{x \in [x_j, x_{j+1}]} |f''(x)|$ and $M = \max_{0 \leq j \leq n-1}\{M_j\}$.

# Some references for numerical quadratures

- Gubner, Gaussian Quadrature and the Eigenvalue Problem `https://gubner.ece.wisc.edu/gaussquad.pdf`,
- Chap 12 in Numerical Computing with Matlab `https://epubs.siam.org/doi/book/10.1137/1.9780898717952`.

# Part 3. Finite difference method for partial differential equations

# Initial value problem and boundary value problem

An initial value problem for a first-order ordinary differential equation is given by

$$y' = f(t, y),\ y(t_0) = y_0.$$

Boundary value problems are differential equations that are subject to boundary conditions. For example,

$$y'' + y = 0$$

with $y(0) = 1$ and $y(\pi/2) = 1$.

# Time-Dependent 1D Heat Equation

Consider the simple example of the time-dependent 1D heat equation

$$\frac{\mathrm{d}}{\mathrm{d}t}u - \frac{\mathrm{d}}{\mathrm{d}x}\left(\frac{\mathrm{d}}{\mathrm{d}x}u\right) = 0, \text{ on } \Omega = [0,1]$$

with the boundary condition

$$u(0,t) = u_a(t),\ u(1,t) = u_b(t) \text{ for all } t \in [0,T]$$

and the initial condition

$$u(x,0) = u_0(x), \text{ for all } x \in [0,1].$$

# Time-Dependent 1D Heat Equation

We need

- Space discretization $\Delta x = h = 1/N$ and grid $x_i = ih$, $i = 0, \ldots, N$,
- Final time $T$, time discretization $\Delta t = k = T/M$, and $t_m = mk$,
- $u_i^m$: approximate solution at $(x_i, t_m)$.

# Difference formula

By using the forward difference, we can compute the derivative of a function $f(x)$ at a point $x_0$.

$$D(f,h) = \frac{f(x_0+h) - f(x_0)}{h} \approx f'(x_0).$$

Let $E(h)$ be the absolute error in the approximation:

$$E(h) = |f'(x_0) - D(f,h)|.$$

# Truncation error in difference formula

The approximation can be derived by using a Taylor series. Expand $f(x_0 + h)$ around $x_0$:

$$f(x_0 + h) = f(x_0) + hf'(x_0) + \frac{h^2}{2} f''(x_0) + \mathcal{O}(h^3).$$

Then,

$$f'(x_0) = \frac{f(x_0 + h) - f(x_0)}{h} - \frac{h}{2} f''(x_0) + \mathcal{O}(h^2).$$

The absolute error in the approximation can be written as

$$D(f, h) = f'(x_0) + \frac{h}{2} f''(x_0) + \cdots .$$

From the leading term of the error we can see that

$$\text{truncation error} \sim Ch \text{ as } h \to 0.$$

# Rounding error in difference formula

Suppose that $f$ is well-conditioned. Then for any evaluation $\hat{f}$ of the function,

$$\hat{f} = f + \delta, \ |\delta| < \epsilon_{\mathrm{m}}.$$

The computed value of $D(f, h)$ is

$$\hat{D}(f, h) = D(f, h) + \frac{\delta_{x_0+h} - \delta_{x_0}}{h}$$

which introduces a rounding error

$$|\text{rounding error}| \leq \frac{\epsilon_{\mathrm{m}} + \epsilon_{\mathrm{m}}}{h} = \frac{2\epsilon_{\mathrm{m}}}{h}$$

# The absolute error in the approximation

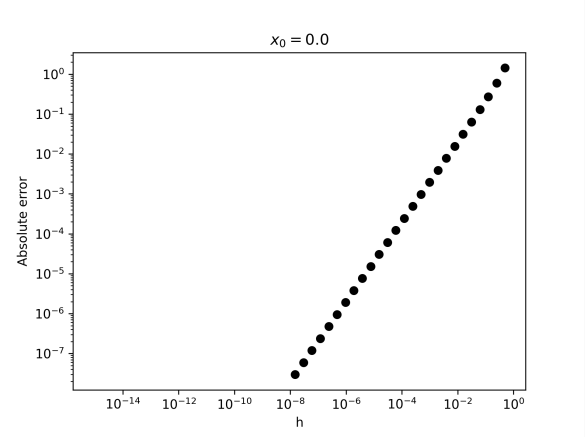
$$E(h) = \text{truncation error} + \text{rounding error}$$
$$\leq Ch + \frac{2\epsilon_{\text{m}}}{h}.$$

Suppose that $f(x) = e^{2x}$.

# The absolute error in the approximation

$$E(h) = \text{truncation error} + \text{rounding error}$$
$$\leq Ch + \frac{2\epsilon_{\text{m}}}{h}.$$

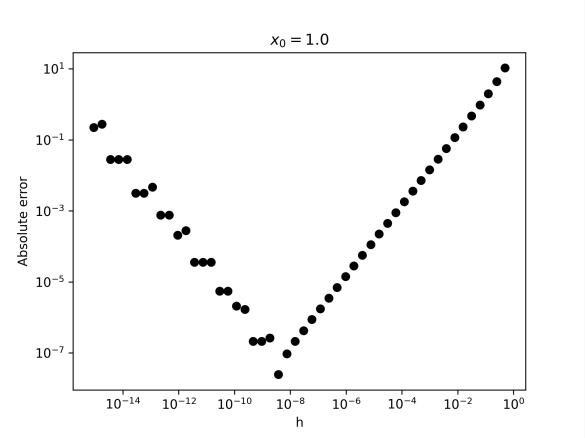
Suppose that $f(x) = e^{2x}$.
Choose $x_0 = 0$.
$f'(x_0) = 2$.



# The absolute error in the approximation

$$E(h) = \text{truncation error} + \text{rounding error}$$
$$\leq Ch + \frac{2\epsilon_{\text{m}}}{h}.$$

Suppose that $f(x) = e^{2x}$.
Choose $x_0 = 1$.
$f'(x_0) \approx 14.7781121979$.



# Difference formula

We can use the difference formula for first-order derivative in $t$

- Forward difference

$$\frac{\mathrm{d}}{\mathrm{d}t}u_i^m = \frac{u_i^{m+1} - u_i^m}{k} + \mathcal{O}(k)$$

- Backward difference

$$\frac{\mathrm{d}}{\mathrm{d}t}u_i^m = \frac{u_i^m - u_i^{m-1}}{k} + \mathcal{O}(k)$$

- Central difference

$$\frac{\mathrm{d}}{\mathrm{d}t}u_i^m = \frac{u_i^{m+1} - u_i^{m-1}}{2k} + \mathcal{O}(k^2)$$

# Time-Dependent 1D Heat Equation

Using three-point central formula for second-order partial derivative in $x$,

$$\frac{d^2}{dx^2}u_i^m = \frac{u_{i-1}^m - 2u_i^m + u_{i+1}^m}{h^2}, \text{ for } i = 1, \ldots, N-1.$$

# Explicit forward Euler method

Then, we can obtain the following linear system

$$\begin{pmatrix} u_1^{m+1} \\ u_2^{m+1} \\ u_3^{m+1} \\ \vdots \\ u_{N-2}^{m+1} \\ u_{N-1}^{m+1} \end{pmatrix} = \begin{pmatrix} u_1^{m} \\ u_2^{m} \\ u_3^{m} \\ \vdots \\ u_{N-2}^{m} \\ u_{N-1}^{m} \end{pmatrix} + \frac{k}{h^2} \begin{pmatrix} -2 & 1 & & & & \\ 1 & -2 & 1 & & & \\ & 1 & -2 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & -2 & 1 \\ & & & & 1 & -2 \end{pmatrix} \begin{pmatrix} u_1^{m} \\ u_2^{m} \\ u_3^{m} \\ \vdots \\ u_{N-2}^{m} \\ u_{N-1}^{m} \end{pmatrix} + \frac{k}{h^2} \begin{pmatrix} u_0^{m} \\ 0 \\ 0 \\ \vdots \\ 0 \\ u_N^{m} \end{pmatrix}$$

# Explicit forward Euler method

Let

$$A = \begin{pmatrix} 2 & -1 & & & & \\ -1 & 2 & -1 & & & \\ & -1 & 2 & -1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & -1 & 2 & -1 \\ & & & & -1 & 2 \end{pmatrix} \in \mathbb{R}^{(N-1)\times(N-1)}.$$

Suppose $u_a(t) = u_b(t) = 0$. Let $\mathbf{u}^{(m)} = (u_1^m, u_1^m, \ldots, u_{N-1}^m)^T$. If the initial condition, $\mathbf{u}^{(0)}$ is the eigenvector of $A$,

$$\mathbf{u}^{(m)} = (1 - s\lambda)^m \mathbf{u}^{(0)}$$

where $s = k/h^2$.

# Stability for explicit forward Euler method

There is no external heat source, $f = 0$, $\mathbf{u}^{(m)}$ would tends to $\mathbf{0}$ with $t$ tending to infinity. That means

$$|1 - s\lambda| < 1.$$

The eigenvalues of $A$, $\lambda_i$ are

$$2 + 2\cos\left(\frac{i\pi}{N}\right), \text{ for } i = 1, \ldots, N-1$$

and its corresponding eigenvectors $\mathbf{v}_i$ are

$$(\mathbf{v}_i)_j = \sin\left(\frac{ij\pi}{N}\right), \text{ for } i, j = 1, \ldots, N-1.$$

# Stability for explicit forward Euler method

We can obtain

$$s < \frac{1}{1 + \cos\left(\frac{i\pi}{N}\right)}$$

for $i = 1, \ldots, N-1$. This gives

$$\frac{k}{h^2} < \frac{1}{2}.$$

We have to choose the time step size $k < \frac{1}{2}h^2$. $\frac{k}{h^2}$ is an important measure for the stability of explicit forward Euler method. It is called the Courant number.
Any other choice of initial condition can be represented as a linear combination of eigenvectors such that, due to the linearity of the equation, all statement hold also for this case.

# Some references for numerical analysis

- Quarteroni, Sacco, and Saleri, Numerical mathematics https://link.springer.com/book/10.1007/b98885,
- Ascher and Greif, A First Course in Numerical Methods https://epubs.siam.org/doi/10.1137/9780898719987,
- Thomas, Numerical Partial Differential Equations: Finite Difference Methods https://link.springer.com/book/10.1007/978-1-4899-7278-1.