

# Numerical Methods

Lev Leontev

February 10, 2023

## Contents

<b>1</b>	<b>Taylor series</b>	<b>1</b>
<b>2</b>	<b>Number representation</b>	<b>5</b>
2.1	Errors . . . . .	5
2.2	Base representation . . . . .	5
2.3	Euclid's algorithm . . . . .	6
2.4	Horner's scheme . . . . .	6
2.5	Floating point representation . . . . .	7

**Definition 1** (Numerical Methods). Numerical Methods are algorithmic approaches to numerically solve mathematical problems. We use it often when it is hard/difficult/impossible to solve analytically.

## 1 Taylor series

Given a function  $f : \mathbb{R} \rightarrow \mathbb{R}$  (that is hard to evaluate for some  $x \in \mathbb{R}$ ), but  $f$  and  $f^{(n)}$  are known for a value  $c$ , which is close to  $x$ . Can we use this information to approximate  $f(x)$ ?

We know values for  $\cos^{(n)}(0)$ .

$$\begin{cases} f(0) = \cos(0) = 1 \\ f'(0) = -\sin(0) = 0 \\ f''(0) = -\cos(0) = -1 \end{cases} \quad \text{for } c = 0$$

Can we get  $\cos(0.1)$  from this?

**Definition 1** (Taylor series). Let  $f : \mathbb{R} \rightarrow \mathbb{R}$ , differentiable infinitely many times at  $c \in \mathbb{R}$ . So we have  $f^{(k)}(c)$ ,  $k = 1, 2, \dots$ . Then the Taylor series of  $f$  at  $c$  is:

$$f(x) \approx f(c) + \frac{f'(c)}{1!}(x-c)^1 + \frac{f''(c)}{2!}(x-c)^2 + \dots = \sum_{k=0}^{\infty} \frac{f^{(k)}(c)}{k!}(x-c)^k$$

**Remark.** Taylor series is a power series.

**Remark.** For  $c = 0$  also known as Maclaurin series

**Remark.** A power series has an interval/radius of convergence. You can only evaluate the series if  $x \in$  interval of convergence.

**Example 1.** What is the Taylor series for  $f(x) = e^x$  at  $c = 0$ ? We have  $f^{(k)}(x) = e^x$ , so  $f^{(k)}(0) = 1$ . Thus:

$$\sum_{k=0}^{\infty} \frac{1}{k!} x^k$$

and the radius of convergence is  $\infty$ .

I.e. for any  $x \in \mathbb{R}$ :

$$e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!}$$

For an algorithm we need a finite amount of terms. For example,

$$e^x \approx \frac{1}{0!}x^0 + \frac{1}{1!}x^1 + \frac{1}{2!}x^2 = 1 + x + \frac{x^2}{2}$$

This is a polynomial!

**Example 2.** Let's calculate Taylor series of a polynomial.

$$f(x) = 4x^2 + 5x + 7, \quad c = 2$$

$$f(2) = 33, \quad f'(2) = 8x + 5 \Big|_{x=2} = 21, \quad f''(2) = 8$$

Taylor series:

$$33 + 21(x-2) + \frac{8}{2}(x-2)^2 = 4x^2 + 5x + 7 = f(x)$$

Taylor series of a polynomial is itself.

**Theorem 1** (Taylor theorem). Let  $f \in C^{n+1}([a, b])$  (i.e.  $f$  is  $(n + 1)$ -times continuously differentiable). Then for any  $x \in [a, b]$  we have that

$$f(x) = \sum_{k=0}^n \frac{f^{(k)}(c)}{k!} (x - c)^k + \frac{f^{(n+1)}(\xi_x)}{(n + 1)!} (x - c)^{n+1}$$

where  $\xi_x$  is a point that depends on  $x$  and which is between  $c$  and  $x$ .

The first sum is called *truncated Taylor series*, the remainder is called *the error*.

**Example.** For  $n = 0$ :

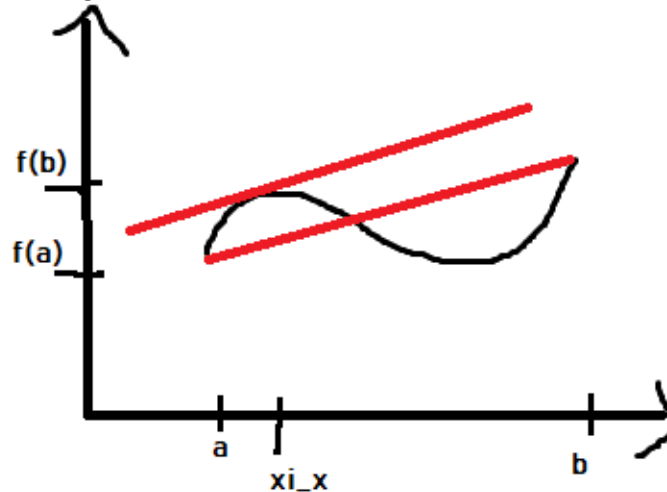
$$f(x) = f(c) + f'(\xi_x)(x - c)$$

Choose  $c = a$ ,  $x = b$ :

$$f(b) = f(a) + f'(\xi_x)(b - a) \iff f'(\xi_x) = \frac{f(b) - f(a)}{b - a}$$

This is the mean value theorem!

Figure 1: Mean value theorem illustration



**Definition 2.** We say that the Taylor series *represents* the function  $f$  at  $x$  if the Taylor series converges at that point, i.e. the remainder tends to zero as  $n \rightarrow \infty$ .

**Example 1.** Back to  $e^x$ :  $f(x) = e^x$ ,  $c = 0$ ,  $\xi_x$  is between  $c$  and  $x$ .

$$e^x = \sum_{k=0}^n \frac{x^k}{k!} + \frac{e^{\xi_x}}{(n + 1)!} x^{n+1}$$

For any  $x \in \mathbb{R}$  we find  $s \in \mathbb{R}_0^+$  ( $\mathbb{R}_0^+$  are all real, positive numbers including 0) so that  $|x| \leq s$ , and  $|\xi_x| \leq s$  because  $\xi_x$  is between  $c$  and  $x$ .



Because  $e^x$  is monotone increasing, we have  $e^{\xi_x} \leq e^s$ , thus

$$\lim_{n \rightarrow \infty} \left| \frac{e^{\xi_x}}{(n+1)!} x^{n+1} \right| \leq \lim_{n \rightarrow \infty} \left| \frac{e^s}{(n+1)!} s^{n+1} \right| = e^s \lim_{n \rightarrow \infty} \frac{s^{n+1}}{(n+1)!} = 0$$

Because  $(n+1)!$  will grow faster than any power of  $s \implies \lim_{n \rightarrow \infty} \left| \frac{e^{\xi_x}}{(n+1)!} x^{n+1} \right| = 0$ .

Thus  $e^x$  is *represented* by its Taylor series.

### Example 2.

$$\begin{aligned} f(x) &= \log(1+x), \quad c=0 \\ f'(x) &= \frac{1}{1+x} = (1+x)^{-1} \\ f''(x) &= -(1+x)^{-2} \\ f'''(x) &= +2(1+x)^{-3} \\ f^{(k)}(x) &= (-1)^{k+1}(k-1)! \frac{1}{(1+x)^k} \end{aligned}$$

So  $f^{(k)}(0) = (-1)^{k-1}(k-1)!$  for  $k \geq 1$ ,  $f(0) = \log(1) = 0$ .

Taylor series:

$$\begin{aligned} f(x) &= \sum_{k=1}^n \frac{(-1)^{k-1}}{k} x^k + \frac{(-1)^n}{n+1} \frac{1}{(1+\xi_x)^{n+1}} \cdot x^{n+1} \quad \left( \frac{n!}{(n+1)!} = \frac{1}{n+1} \right) \\ E_n(x) &= \frac{(-1)^n}{n+1} \frac{1}{(1+\xi_x)^{n+1}} \cdot x^{n+1} \text{ --- the remainder} \end{aligned}$$

Question: for which  $x$  does  $\lim_{n \rightarrow \infty} E_n(x) = 0$ ?

$$\lim_{n \rightarrow \infty} E_n(x) = \lim_{n \rightarrow \infty} \frac{(-1)^n}{n+1} \left( \frac{x}{\xi_x + 1} \right)^{n+1} \text{ for } \xi_x \in (c, x) \quad (c=0)$$

Such a limit converges to 0, if the fraction is less than 1.

$$0 < \frac{x}{\xi_x + 1} < 1 \iff x < \xi_x + 1 \iff x - \xi_x < 1 \text{ with } \xi_x \in (0, x) \iff x \leq 1$$

**Consequence.**  $\lim_{n \rightarrow \infty} E_n(x) = 0$  if  $0 < x \leq 1$ . This means that the Taylor series represents  $\log(x+1)$  for  $x \in [0, 1]$ . We can extend this to show  $x \in (-1, 1]$ .

**Example 3.** Let's compute  $\cos(0.1)$ . Let's approximate it with Taylor series with  $c=0$  (around zero).

$$\cos(x) = 1 - \frac{x^2}{2} + \frac{x^4}{4!} - \frac{x^6}{6!} \pm \dots + \text{remainder}$$

**Consequence.**

$$\left| \cos(x) - \sum_{k=0}^n (-1)^k \frac{x^{2k}}{(2k)!} \right| = \left| (-1)^{n+1} \cos(\xi_x) \frac{x^{2(n+1)}}{(2(n+1))!} \right| \leq \frac{0.1^{2(n+1)}}{2(n+1)!} \xrightarrow{n \rightarrow \infty} 0$$

$n$	Taylor polynomial	$ error  \leq$
0	1	$\frac{(0.1)^2}{2} = 0.0005$
1	0.995	$\frac{0.0001}{24}$
2	0.99500416	$\frac{0.000001}{6!}$

Error depends on choice of  $|x - c|$  and  $n$ .

**Example 4.** Compute  $\log(2)$  using  $f(x) = \log(x + 1)$

$$\log(2) = 1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \frac{1}{5} - \frac{1}{6} + \dots$$

Keeping 8 terms (until  $n = 8$ ) we get  $\log(2) \approx 0.63452$ , the actual solution is  $\log(2) = 0.693147$ . Not so accurate. Can we improve?

We can use Taylor series of  $\log\left(\frac{1+x}{1-x}\right)$  instead, since  $\log\left(\frac{1+x}{1-x}\right) = \log(1+x) - \log(1-x)$ . We choose  $x = \frac{1}{3}$  instead of  $x = 1$ . Since  $x$  is closer to zero, both of the logarithms converge quicker.

$$\left( \log\left(\frac{1+1/3}{1-1/3}\right) = \log(2) \right)$$

We then get

$$\log(2) = 2 \cdot \left( \frac{1}{3} + \frac{1}{3^3 \cdot 5} + \dots \right)$$

We only need 4 terms to get  $\log 2 \approx 0.69313$ .

**Theorem 2** (Reformulation of Taylor's theorem).  $f \in C^{n+1}([a, b])$ . We change  $c$  to  $x$  and the old  $x$  to  $x + h$  from previous version  $\implies$  get for  $x, x + h \in [a, b]$ :

$$f(x + h) = \sum_{k=0}^n \frac{f^{(k)}(x)}{k!} h^k + \frac{f^{(n+1)}(\xi_x)}{(n+1)!} h^{n+1} \text{ where } \xi_x \in (x, x + h), h > 0$$

We can write error term as

$$f(x + h) - \sum_{k=0}^n \frac{f^{(k)}(x)}{k!} h^k = \mathcal{O}(h^{n+1})$$

**Remark.** Let's recall what the  $\mathcal{O}$ -notation means.  $a(h) = \mathcal{O}(b(h))$  if  $\exists c > 0$  such that  $\frac{a(h)}{b(h)} \leq c$  as  $h \rightarrow 0$ . So, for  $n = 1$  the error decreases with  $h^2$  (quadratic convergence).  $n = 2$ : error decreases cubically, i.e.  $h^3$ , etc.

### Summary of Taylor series:

- Problem: Evaluate  $f(x)$  with a given error bound.
- Required:  $f \in C^{n+1}$ , values of derivatives  $f^{(k)}(a)$ .
- Check interval of convergence: does Taylor series expansion work?
- Estimate the maximum error for  $n$  terms of the Taylor polynomial.
- Choose  $n$ , such that the error bound is low enough.
- Evaluate the Taylor polynomial.

## 2 Number representation

### 2.1 Errors

There are different error types:

1. An error in data (partly due to roundoff).
2. Roundoff errors (during computation). For example, multiplication increases the amount of needed significant digits, and we can't store them all on a computer.
3. Truncation error, that is inherent to numeric methods. For example, if we take a finite number of terms in our Taylor series.

**Definition 1.** Let  $\tilde{a}$  be an approximation of  $a$ . Then  $|\tilde{a} - a|$  is the *absolute error*, and  $|\frac{\tilde{a}-a}{a}|$  is the *relative error*. The *error bound* is the magnitude of admissible error.

**Example.** 0.00123 with error  $0.000004 = 0.4 \cdot 10^{-5}$ . The error is below  $\frac{1}{2} \cdot 10^{-t}$  with  $t = 5$ , so there's 5 correct digits and 3 significant digits (the number of non-leading zeros).

**Example.** 0.00123 with error  $0.000006 = 0.06 \cdot 10^{-4} > \frac{1}{2} \cdot 10^{-5}$ . Only has 2 significant digits, because we have to round the error up.

**Theorem 1.** In addition/subtraction the bounds for absolute errors are added, in multiplication/division the relative errors are added.

**Example.** Solve  $x^2 - 56x + 1 = 0$ :

$$x = 28 - \sqrt{783} \approx 28 - 27.982 \text{ (5 significant digits)} = 0.018 \pm \frac{1}{2} \cdot 10^{-3}$$

We end up with 2 significant digits in the answer, despite that we used to have 5. That's why computers use floating point numbers, as leading zeros are bad.

**Definition 2** (Error propagation). If  $y(x)$  is smooth, than the derivative  $|y(x)'|$  can be interpreted as the sensitivity of  $y(x)$  to errors in  $x$ . We can generalize this to functions of multiple variables:

$$|\Delta y| \leq \sum \left| \frac{\partial y}{\partial x_i} \right| \cdot |\Delta x_i|, \text{ where } \Delta y = \tilde{y} - y, \Delta x_i = \tilde{x}_i - x_i$$

This is an empirical inequality that is only valid for small  $\Delta x_i$ . It is used a lot in physics.

### 2.2 Base representation

**Definition 3** (Base representation). Every number  $x \in \mathbb{N}$  can be written in the following form as a unique expansion with respect to the base  $b$ , where  $b \in \mathbb{N} \setminus \{1\}$ :

$$x = a_0 b^0 + a_1 b^1 + a_2 b^2 + \dots + a_n b^n = \sum_{i=0}^n a_i b^i$$

$$a \in \mathbb{N}_0, a_i < b, a_i \in \{0, \dots, b-1\}$$

Here  $b$  is called the *base*,  $a_i$  are called the *digits*. Humans usually use base 10. But, for example, computers can use base 2.

For a real number  $x \in \mathbb{R}$  we can write:

$$x = \sum_{i=0}^n a_i b^i + \sum_{i=1}^{\infty} a_{-i} b^{-i}$$

**Example.**  $b = 2$  :  $1011 = 1 \cdot 2^0 + 1 \cdot 2^1 + 0 \cdot 2^2 + 1 \cdot 2^3 = (11)_{10}$

There are different algorithms that convert number systems.

## 2.3 Euclid's algorithm

Euclid's algorithm converts  $(x)_{10}$  to  $(y)_b$ .

1. Input  $(x)_{10}$ .
2. Determine the smallest  $n$ , such that  $x < b^{n+1}$ .
3. For  $i = n$  to 0 do:

$$\begin{aligned} a_i &:= x \operatorname{div} b^i \text{ (integer division)} \\ x &:= x \operatorname{mod} b^i \text{ (the remainder)} \end{aligned}$$

4. Output result  $a_n a_{n-1} a_{n-2} \dots a_0 = (y)_b$ .

**Example.** 1.  $(x)_{10} = (13)_{10} \rightarrow (y)_2$

2.  $n = 3$  since  $13 < 2^4$ .
- 3.

$$\begin{aligned} i = 3 : a_3 &= 13 \operatorname{div} 2^3 = 1, \quad x = 13 \operatorname{mod} 2^3 = 5 \\ i = 2 : a_2 &= 5 \operatorname{div} 2^2 = 1, \quad x = 5 \operatorname{mod} 2^2 = 1 \\ i = 1 : a_1 &= 1 \operatorname{div} 2^1 = 0, \quad x = 1 \operatorname{mod} 2^1 = 1 \\ i = 0 : a_0 &= 1 \operatorname{div} 2^0 = 1, \quad x = 1 \operatorname{mod} 2^0 = 0 \end{aligned}$$

4. Output:  $(1101)_2 = (13)_{10}$ .

Two problems of the Euclid's algorithm:

1. Step 2 is inefficient
2. Division by large numbers can be problematic.

## 2.4 Horner's scheme

Horner's scheme is a more efficient algorithm. The idea is to represent the number as follows:

$$(a_n a_{n-1} \dots a_0)_b = a_0 + b(a_1 + b(a_2 + b(a_3 + \dots + b(a_n)))) \dots$$

The algorithm is the following:

1. Input  $(x)_{10}$ .
2.  $i := 0$ .

3. While  $x > 0$  do:

$$\begin{aligned} a_i &:= x \bmod b \\ x &:= x \operatorname{div} b \\ i &:= i + 1 \end{aligned}$$

4. Output result  $a_n a_{n-1} a_{n-2} \dots a_0 = (y)_b$ .

**Remark.** The algorithm is very similar to the Euclid's algorithm — the difference is that we execute it in reverse. We no longer have divisions by large numbers, and thus it runs faster.

### General remarks:

- A number with simple representation in one base may be complicated to represent in another base. For example,  $(0.1)_{10} = (0.0001100110011\dots)_2$ .
- Base 2 is called *binary*, base 8 is *octal*, base 16 is *hexadecimal*.
- To convert from a base  $b$  to base 10 we can just perform the following computation:

$$(42)_8 = 4 \cdot 8^1 + 2 \cdot 8^0 = (34)_{10}$$

- Conversion 2 and 8.  $8 = 2^3$ : three consecutive bits represent one octal digit, e.g.

$$(551.624)_8 = (101101001.110010100)_2$$

- Conversion 2 and  $16 = 2^4$ : just four bits to one hexadecimal digit.
- Horner's scheme algorithm does not need estimate of  $n$  and does not divide by large numbers.
- It is applicable to real numbers, but one needs a criterion to stop if the representation with the new base is infinite.
- On computers we only have finite precision (the number of digits/bits).

## 2.5 Floating point representation

**Definition 4.** Normalized floating point representation with respect to a base  $b$  stores any number  $x$  as follows:

$$x = 0.a_1 a_2 \dots a_k \cdot b^n \text{ where } a_i \in \{0, 1, \dots, b-1\}$$

$a_i$  are called *digits*,  $k$  is called *precision*,  $n$  is called *exponent*,  $a_1 \dots a_k$  is called *mantissa*,  $a_1 \neq 0$  is called *normalization*, which makes the representation unique.

**Remark.** Leading zeros are essentially a waste of space. That's why floating points are useful when you add/subtract/divide numbers — as you're able to move the decimal point.

**Example 1.** Base 10:  $32.213 = 0.32213 \cdot 10^2$ .

Base 2:  $x = \pm 0.b_1 b_2 \dots b_k \cdot 2^n$  We need another bit to define the sign of the number.

**Example 2.** For *single precision* floating-point numbers: 4 bytes = 32 bits.

- 1 bit sign mantissa.



- 1 bit sign exponent.
- 7 bits for exponent (integer).
- 23 bits for mantissa (24 effectively, since the first digit is always 1).

7 bits allow for the largest exponent of 127, i.e.  $2^{127} \approx 10^{38}$ . Anything above that is infinity. So, we have the range of  $10^{-38} < |x| < 10^{38}$ .

Since  $2^{-24} \approx 10^{-7}$ , we can represent 7 significant digits.

We can have a better representation, e.g. with *double precision* (8 bytes).

Issues with floating-point numbers:

- Adding numbers is not commutative, i.e.  $x + y$  does not necessarily equal to  $y + x$ .
- It's not associative, i.e.  $(x + y) + z$  does not necessarily equal to  $x + (y + z)$ . You usually try to add small numbers together first, in hopes that they will become big enough to become significant.

**Example 1.** Take  $x = y = 0.00000033$ ,  $z = 0.00000034$ ,  $w = 1.00000000$ . We compute  $x + y + z + w$ , and in that order we get 1.000001. Reversed order gets 1.000000 with base 10. Why is that? That's because we only have 7 significant digits.

**Example 2.**

$$\sum_{1}^{10^7} 1 + 10^7 = 1 + \dots + 1 + 10^7$$

The order of summation will make a difference. You will either get  $10^7$  or  $2 \cdot 10^7$ , because  $1 + 1 = 2$ , but  $1 + 10^7 = 10^7$  due to roundoff.

**Consequence.** Avoid adding numbers of different order of magnitude. Add numbers in increasing order of their size.

**Example 3.** Compute  $x - \sin x$  for  $x$  close to 0, e.g.  $x = 1/15$ . Assume  $k = 10$  precision.

$$\begin{aligned} x &= 0.6666666667 \cdot 10^{-1} \\ \sin x &= 0.6661729492 \cdot 10^{-1} \\ x - \sin x &= 0.0004937175 \cdot 10^{-1} = 0.4937175000 \cdot 10^{-4} \end{aligned}$$

Notice the three zeros at the end — that is a sign of a precision loss (unless the number actually ends with zeros).

**Consequence.** Avoid subtracting numbers of similar size, because it leads to a loss of precision.

A potential solution in this case would be to use Taylor series expansion:

$$\begin{aligned} \sin x &= x - \frac{x^3}{3!} + \frac{x^5}{5!} - \dots \\ x - \sin x &= +\frac{x^3}{3!} - \frac{x^5}{5!} + \dots \end{aligned}$$

Using 3 terms we get:  $0.4937174328 \cdot 10^{-4}$ . The error of the Taylor series with 3 terms will be  $\leq 10^{-13}$  (which is needed for “full” precision for  $0 \dots \cdot 10^{-4}$ ).

**Theorem 2.** Let  $x, y$  be two normalized floating point numbers with  $x > y > 0$  and base  $b = 2$ . If there exist  $p, q \in \mathbb{N}_0$  such that

$$2^{-p} \leq 1 - \frac{y}{x} \leq 2^{-q}$$

Then at most  $p$  and at least  $q$  significant bits (digits base 2) are lost during subtraction.