MA 214: Introduction to numerical analysis
Lecture 1

Shripad M. Garge.
IIT Bombay

(shripad@math.iitb.ac.in)

2021-2022

Welcome

It is a pleasure to welcome you to this course on numerical analysis.

Question

What is **numerical analysis***?*

This question does not have an easy answer.

Hopefully you will be able to give your own answer to this question after this course is over.

This is so because the answer to this question depends on how you plan to use *numerical analysis* after you are done with this semester.

Introduction

For now, numerical analysis is the study of various methods

- to solve various kinds of equations,
  - differential equations,
  - systems of linear equations,
  - $f(x) = \alpha$.
- to approximate functions and
- to study the corresponding errors.

Whenever solutions exist, we will also focus on constructing (approximate) solutions.

Let us start with an example.

The number e

We all know the definition of the number e, namely,

$$e = \lim_{n\to\infty} (1 + 1/n)^n.$$

But can you compute the value of e which is correct for the first 10 digits?

$$2.718281828$$

first 20 digits?

$$2.7182818284590452353$$

how would one do it?

Approximating e

There is the exponential function, $\exp(x)$, whose value at the point $x = 1$ is the value e.

This exponential function is one of the best known functions to us, it is infinitely many times differentiable, so we can approximate it using Taylor's theorem to any degree of precision we want.

$$f(x) = f(a) + f'(a)(x-a) + \cdots + \frac{f^{(k)}(a)}{k!}(x-a)^k$$
$$+\frac{f^{(k+1)}(c)}{(k+1)!}(x-a)^{k+1}.$$

Approximating e

It tells us that

$$e = exp(1) = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \cdots + \frac{1}{n!} + \frac{e^c}{(n+1)!}$$

where c is some real number between 0 and 1.

Here, the error term at the n-th approximation is the term $e^c/(n+1)!$.

We know that between 0 and 1, the value of e^c is less than 3 so we can compute the n where $e^c/(n+1)!$ is less than the prescribed error.

Approximating e

For instance, if we want the error to be less than 10^{-10} then we must find an n such that $e^c/(n+1)! < 10^{-10}$.

This can be achieved if we find an n with $3/(n+1)! < 10^{-10}$.

The smallest such n happens to be 13.

Now this calculation can be done on a calculator.

You might as well ask why we did not do the original calculation on the calculators?

Calculators

The calculator knows the value of e to a **certain** accuracy and our point was to find the value up to **our** accuracy.

Note that we actually found a "function" that approximates the exponential function and used it to approximate e to our accuracy.

We can use the function to approximate any e^{α} up to the given accuracy.

Aspects of numerical analysis

This straightaway gives two different aspects of numerical analysis.

- The theory behind the calculation.

In our case, it was knowing the properties of the exponential function and the Taylor's theorem.

- The computation, which will often be done by calculators.

We will be focussing slightly more on the theory part and some light computations.

The heavier computations will be done when you actually use numerical analysis in your own disciplines.

Programming

Programming is a different ball game altogether.

You have had a full semester course on programming by competent teachers and I will add little to that.

We will also have a shortage of time if we spend any time on programming.

I do not mean to discourage you from programming.

If you are interested in it, then please do it by all means and you will discover that you learn a lot by doing things for pleasure than doing them for marks.

Representing a number

We will be doing calculations with numbers, so let us take a quick stock of how the numbers are represented.

We routinely use the representation at base 10, so we write

$$2022 = (2 \times 10^3) + (0 \times 10^2) + (2 \times 10^1) + (2 \times 10^0).$$

If we were to write 2022 in base 2 it would be 11111100110.

There are easy methods to transform the representation of a number in base 'a' to that in base 'b' and you should look it up as an interesting homework exercise.

How calculators see numbers

Let us do a simple calculation to understand how calculators (and computers) understand numbers.

Let us solve the equation:

$$ax^2 + bx + c = 0$$

for $a = 1$, $b = -(10^4 + 10^{-4})$ and $c = 1$.

The formula tells us that the solutions are

$$x_1 = 10^4 \quad \text{and} \quad x_2 = 10^{-4}$$

however a calculator with the percision of seven decimal points will give $x_1 = 10^4$ and $x_2 = 0$.

How calculators see numbers

The error in the second solution is 100 percent!

This is because of the round-off error.

We will need to see how we deal with such cases but it is also important to have some cross-calculations to check the answers obtained using machines.

This is where some special knowledge of the problem, *the theory component*, will be useful.

However large the capacity of a machine may be, it is still finite!

MA 214: Introduction to numerical analysis
Lecture 2

Shripad M. Garge.
IIT Bombay

(shripad@math.iitb.ac.in)

2021-2022

Finite digit arithmetic

We ended our last lecture by observing that the computer world is a world of finite digit arithmetic.

Let us try to understand it further.

When we do arithmetic, we allow for infinitely many digits.

For instance, we understand that

$$\frac{1}{3} = 0.33333333\cdots = 0.\bar{3}$$

or that the decimal expansion of $\sqrt{3}$ is infinite, it does not terminate.

Finite digit arithmetic

In the computational world, however, each representable number has only a fixed and finite number of digits.

This means, for example, that only rational numbers – and not even all of these – can be represented exactly.

Since $\sqrt{3}$ is not rational, it is given an approximate representation, one whose square will not be precisely 3, although it will likely be sufficiently close to 3 to be acceptable in most situations.

In most cases, then, this machine arithmetic is satisfactory and passes without notice or concern, but at times problems arise because of this discrepancy.

Round-off error

The error that is produced when a calculator or computer is used to perform calculations is called round-off error.

It occurs because the calculations are performed with only approximate representations of the actual numbers.

In a computer, only a relatively small subset of the real numbers is used for the representation of all the real numbers.

This subset contains only rational numbers, both positive and negative, and stores the fractional part, together with an exponential part.

Representing real numbers

A 64-bit (binary digit) representation is used for a real number.

The first bit is a sign indicator, denoted by s.

This is followed by an 11-bit exponent, c, called the characteristic, and a 52-bit binary fraction, f, called the mantissa.

The base for the exponent is 2.

The exponent of 11 binary digits gives a range of 0 to $2^{11} - 1 = 2047$.

Representing real numbers

However, using only positive integers for the exponent would not permit an adequate representation of numbers with small magnitude.

To ensure that numbers with small magnitude are equally representable, 1023 is subtracted from the characteristic, so the range of the exponent is actually from -1023 to 1024.

Thus, the system gives a floating-point number of the form

$$(-1)^s 2^{c-1023}(1+f).$$

Since 52 binary digits correspond to between 16 and 17 decimal digits, we can assume that a number represented in this system has at least 16 decimal digits of precision.

Example

Consider the machine number

0 10000000011 1011100100010000000000000000000000000000000000000000.

The leftmost bit is $s = 0$, which indicates that the number is positive.

The next 11 bits, 10000000011, give the characteristic and are equivalent to the decimal number

$$c = 1 \cdot 2^{10} + 0 \cdot 2^9 + \cdots + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 1024 + 2 + 1 = 1027.$$

The exponential part of the number is, therefore,

$$2^{1027-1023} = 2^4.$$

Example

0 10000000011 1011100100010000000000000000000000000000000000000000.

The final 52 bits specify that the mantissa is

$$f = 1\cdot\left(\frac{1}{2}\right)^{1} + 1\cdot\left(\frac{1}{2}\right)^{3} + 1\cdot\left(\frac{1}{2}\right)^{4} + 1\cdot\left(\frac{1}{2}\right)^{5} + 1\cdot\left(\frac{1}{2}\right)^{8} + 1\cdot\left(\frac{1}{2}\right)^{12}.$$

Thus, the represented number is

$$(-1)^s 2^{c-1023}(1+f)$$

$$= (-1)^0 2^{1027-1023}\left(1 + \left(\frac{1}{2} + \frac{1}{8} + \frac{1}{16} + \frac{1}{32} + \frac{1}{256} + \frac{1}{4096}\right)\right)$$

$$= 27.56640625.$$

The smallest and the largest representable numbers

The smallest positive number that can be represented is with $s = 0$, $c = 1$ and $f = 0$, so it is

$$2^{-1022} \cdot (1 + 0) \approx 0.22251 \times 10^{-307}.$$

Numbers occurring in calculations that have a magnitude less than this number result in underflow and are generally set to zero.

Numbers occurring in calculations that have a magnitude higher than

$$2^{1023} \cdot (2 - 2^{-52})$$

result in overflow and typically cause the computation to stop.

Floating-point representation

To understand how the computer does the calculations, we will use a similar method to write numbers but with familiar decimal representations instead of the binary ones.

We will use numbers of the form

$$\pm\, 0.d_1 d_2 \ldots d_k \times 10^n, \qquad 1 \leqslant d_1 \leqslant 9, \qquad 0 \leqslant d_i \leqslant 9$$

for each $i = 2, \ldots, k$.

Floating-point representation

For instance, if a positive number y has decimal expansion

$$y = 0.d_1 d_2 \dots d_k d_{k+1} \dots \ \times 10^n$$

with $1 \leqslant d_1 \leqslant 9$ then there are two ways to get the corresponding floating-point representations.

Chopping: we simply chop off the digits $d_{k+1} d_{k+2} \dots$ and write

$$fl(y) = 0.d_1 d_2 \dots d_k \ \times 10^n$$

Rounding: we add $5 \times 10^{n-(k+1)}$ to y and then chop the result to obtain

$$fl(y) = 0.\delta_1 \delta_2 \dots \delta_k \ \times 10^n$$

The five digit chopping and rounding values of π are respectively

$$0.31415 \times 10^1 \quad \text{and} \quad 0.31416 \times 10^1.$$

The errors

Now, this approximate way of writing numbers is bound to create errors.

If p is a real number and if p^* is its approximation then the absolute error is $|p - p^*|$ while the relative error is $\dfrac{|p - p^*|}{p}$ whenever $p \neq 0$.

If $p = 0.1000 \times 10^n$ and if $p^* = 0.1100 \times 10^n$ then the absolute error is $0.1 \times 10^{n-1}$ while the relative error is simply 10^{-1}.

The absolute error can be misleading while the relative error is more meaningful as it takes into account the size of the number p.

Significant digits

We say that the number p^* approximates p to t significant digits if t is the largest non-negative integer for which

$$\frac{|p - p^*|}{p} < 5 \times 10^{-t}.$$

The following table lists the lub of the absolute error when p^* approximates p to four significant digits.

p	0.1	0.5	100	1000	9990	10000
$\max\|p - p^*\|$	0.00005	0.00025	0.05	0.5	4.995	5.

Finite digit arithmetic

The arithmetic in machines is defined by the following:

$$x \oplus y := fl\big(fl(x) + fl(y)\big), \quad x \ominus y := fl\big(fl(x) - fl(y)\big),$$

$$x \otimes y := fl\big(fl(x) \times fl(y)\big), \quad x \oplus\!\!\!\!\div\, y := fl\big(fl(x) \div fl(y)\big).$$

This arithmetic corresponds to performing exact arithmetic on the floating-point representations of x and y and then converting the exact result to its floating-point representation.

We will see some computations of this sort during our tutorials and will also see some common reasons for the occurrances of errors.

MA 214: Introduction to numerical analysis
Lecture 3

Shripad M. Garge.
IIT Bombay

(shripad@math.iitb.ac.in)

2021-2022

Finite digit arithmetic

We have seen the chopping and the rounding methods of writing k-digit approximations of real numbers.

The machine arithmetic is defined by:

$$x \oplus y := fl\big(fl(x) + fl(y)\big), \qquad x \ominus y := fl\big(fl(x) - fl(y)\big),$$

$$x \otimes y := fl\big(fl(x) \times fl(y)\big), \qquad x \oplus\!\!\!\!\div\, y := fl\big(fl(x) \div fl(y)\big).$$

Today, we will see some examples of how errors creep in during these calculations.

Cancellation of nearly equal numbers

One of the major sources of errors is cancellation of two nearly equal numbers.

Let

$$fl(x) = 0.d_1 d_2 \dots d_p \alpha_{p+1} \alpha_{p+2} \dots \alpha_k \times 10^n$$

and

$$fl(y) = 0.d_1 d_2 \dots d_p \beta_{p+1} \beta_{p+2} \dots \beta_k \times 10^n.$$

Then

$$fl\big(fl(x) - fl(y)\big) = 0.\gamma_{p+1} \gamma_{p+2} \dots \gamma_k \times 10^{n-p}.$$

Thus, the difference $x \ominus y$ has $k - p$ digits of significance compared to the k digits of significance of x and y.

Any further calculation with $x \ominus y$ will have no more than $k - p$ significant digits.

Cancellation of nearly equal numbers

Let us see an example.

Let $x = \frac{1}{3}$ and $y = 0.3333 \times 10^0$.

In five digit arithmetic,

$$fl(x) = 0.33333 \times 10^0, \qquad fl(y) = 0.33330 \times 10^0$$

Hence $x \ominus y = 0.00003 \times 10^0 = 0.3 \times 10^{-4}$.

The absolute error is $0.00001/3$ and the relative error is $1/10$ which is reasonably large.

Cancellation of nearly equal numbers

Another example of the subtraction error is as follows:

Consider the function $f(x) = 1 - \cos x$.

If we take $x = 1.2 \times 10^{-5}$ then $\cos x = 0.999\ 999\ 999\ 9$, rounded to 10 digits, so $1 - \cos x = 0.000\ 000\ 000\ 1$.

Even after starting with 10 significant digits, we get only one significant digit for $1 - \cos x$.

Once again the relative error is reasonably big, in fact, it is more than 1.

Division by small numbers

Another way the errors creep in is when we divide by numbers of small magnitude or multiply by numbers of large magnitude.

Let a be a real number and $fl(a) = a + \epsilon$ be its floating point representation with an error ϵ.

If we now divide a by $b = 10^{-n}$ then the machine arithmetic is

$$a \oplus\!\!\!\!\div\, b = fl\left(\frac{fl(a)}{fl(b)}\right) = (a + \epsilon) \times 10^n.$$

The absolute error in this calculation is now $|\epsilon| \times 10^n$.

Errors

Consider the quadratic equation $ax^2 + bx + c = 0$ whose roots are given by the formulae

$$\frac{-b + \sqrt{b^2 - 4ac}}{2a}, \qquad \frac{-b - \sqrt{b^2 - 4ac}}{2a}.$$

Here if b^2 is large compared to $4ac$ then the machine is likely to treat $4ac$ as zero.

This is another way the error can creep in during the cancellations.

We have already seen one such example during our very first lecture.

Errors

One way to get around this problem is to rationalize the formula:

$$\frac{-b+\sqrt{b^2-4ac}}{2a} = \frac{-b+\sqrt{b^2-4ac}}{2a} \times \frac{(-b-\sqrt{b^2-4ac})}{(-b-\sqrt{b^2-4ac})}$$

which gives an alternate formula

$$\frac{-2c}{b+\sqrt{b^2-4ac}}.$$

A similar alternate formula for the other root is

$$\frac{-2c}{b-\sqrt{b^2-4ac}}.$$

Errors

However, one needs to be careful before using these alternate formulae.

If b is too big compared to the product $4ac$ then the alternate formula will result in big error, as we will be dividing by the difference of two nearly equal numbers. The sign of b determines the formula that we should use.

Thus, the cancellation error can come in two major ways:

1. when we cancel two nearly equal numbers, and
2. when we subtract a small number from a big number.

Errors propagate

Another important point to note is that the errors propagate.

Once an error is introduced in a calculation, any further operation is likely to contain the same or a higher error.

This happens when we compute values of certain functions, even the well-behaved functions like the polynomial functions.

Let us compute the value of the polynomial $x^3 + x^2 + x + 1$ at the point $x = 1.876$ using two-digit arithmetic.

Computing values of functions

We want to compute $x^3 + x^2 + x + 1$ at $x = 1.876$.

	x	x^2	x^3	$f(x)$
Exact	1.876	3.519376	6.602349376	12.997725376
2-digit chopping	1.87	3.49	6.52	12.88
2-digit rounding	1.88	3.53	6.64	13.05

In this computation, we have not used any of the error-prone operations. A simple finite digit arithmetic has produced such an error.

It is therefore easy to imagine what the error-prone operations can do.

Can we avoid errors?

When we use finite arithmetic, the errors are inevitable, as we saw from the simple example above.

There are some measures that we can take to try to avoid errors or, at the least, to minimise them. But these measures depend on the case at hand, they vary as the cases vary.

For instance, a general degree 3 polynomial can be written as

$$ax^3 + bx^2 + cx + d = x(x(ax + b) + c) + d$$

as a nested polynomial. Computations done using this form will typically have a smaller error as the number of operations has reduced.

The first theme is complete

This brings us to the end of the first theme of our course:

Machine arithmetic.

From our next lecture, we will begin the next theme:

Equations in one variable.

MA 214: Introduction to numerical analysis
Lecture 4

Shripad M. Garge.
IIT Bombay

(shripad@math.iitb.ac.in)

2021-2022

Roots of equations

We now begin to discuss methods to find roots of equations in one variable. So, given a function f, we are interested in finding x such that

$$f(x) = 0.$$

There are many reasons to be interested in finding roots of equations.

It seems like a special problem but it is equivalent to solving for $f(x) = \alpha$.

We begin with the most basic method to find a root of an equation, the bisection method.

Bisection method

Intermediate Value Theorem

If $f : [a, b] \to \mathbb{R}$ is a continuous function with $f(a) \cdot f(b) < 0$ then there is a $c \in [a, b]$ such that $f(c) = 0$.

The bisection method works with functions f satisfying the condition given in the above theorem and finds a sequence whose limit is a root for f.

The method calls for a repeated halving (or bisecting) of subintervals of $[a, b]$ and, at each step, locating the half containing a root of f.

That explains the name.

Bisection method

To begin, set $a_1 = a$ and $b_1 = b$, and let p_1 be the midpoint of $[a, b]$; that is,

$$p_1 = \frac{a_1 + b_1}{2}.$$

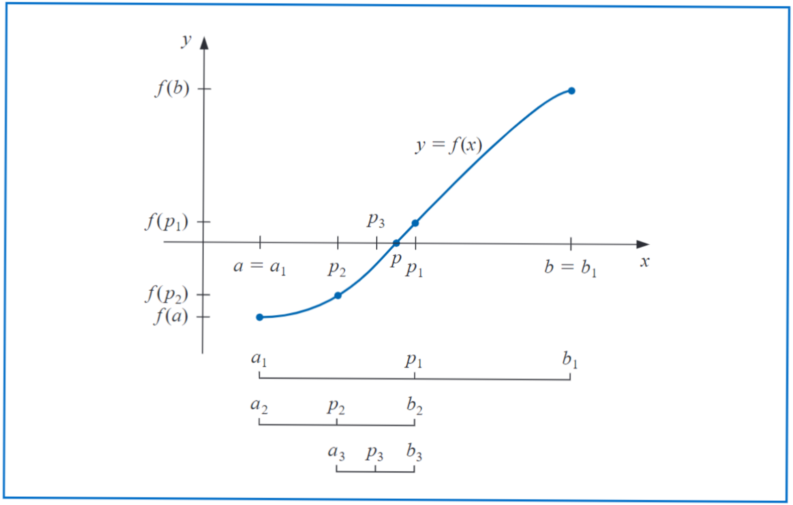
• If $f(p_1) = 0$, then p_1 is a root of f and we are done.

• If $f(p_1) \neq 0$ then $f(p_1)$ has the same sign as that of $f(a_1)$ or $f(b_1)$.

- If $f(p_1)f(a_1) > 0$ then the subinterval $[p_1, b_1]$ contains a root of f, we then set $a_2 = p_1$ and $b_2 = b_1$.
- If $f(p_1)f(b_1) > 0$ then the subinterval $[a_1, p_1]$ contains a root of f, we then set $a_2 = a_1$ and $b_2 = p_1$.

We then repeat the process.

Bisection method



Finding the initial interval

Given a function f whose root is to be found, we begin by finding an interval satisfying the hypothesis of the intermediate value theorem.

Since each step reduces the length of the interval by a factor of two, it is advantageous to find the initial interval as small as possible.

For instance, for the function $f(x) = 2x^3 - x^2 + x - 1$ we have

$$f(-4) \cdot f(4) < 0, \quad \text{and} \quad f(0) \cdot f(1) < 0$$

so any of the intervals $[-4, 4]$ and $[0, 1]$ could be taken as the initial interval but it is better to choose the latter one.

Terminating the bisection method

The sequence $\{p_i\}$ will converge to a root of f.

But it may happen that none of the p_i is a root.

For instance, the root may be an irrational number and a_1, b_1 may be rational numbers. Then all p_i will be rational numbers and none of them will be a root of f.

Take, for instance, $f(x) = x^2 - 2$ defined on $[1, 2]$.

We need to therefore decide when to stop the bisection method and consider the resulting p_N as the (approximate) root of f.

Terminating the bisection method

There are three criteria to decide to stop the bisection process.

- $|f(p_n)| < \epsilon$,
- $|p_n - p_{n-1}| < \epsilon$,
- $p_n \neq 0$ and $\dfrac{|p_n - p_{n-1}|}{|p_n|} < \epsilon$.

Terminating the bisection method

The first criteria, $|f(p_n)| < \epsilon$, is not good because it may take a long time to achieve it, or that it may be achieved easily and yet p_n may be significantly many steps away from a root of f.

The second criterion, $|p_n - p_{n-1}| < \epsilon$, is not good because it does not take the function f into account.

The corresponding n can simply be computed once we know $|a - b|$. As a result, while p_n and p_{n-1} may be close enough the actual root may still be many steps away.

The last criterion looks enticing because it mimics the relative error but again it does not take the function f into account.

It is common to use the second criterion.

An example

Let $f(x) = x^3 + 4x^2 - 10$.

We observe that $f(1) = -5$ and $f(2) = 14$, so there is a root of f in the interval $[1, 2]$ by the intermediate value theorem.

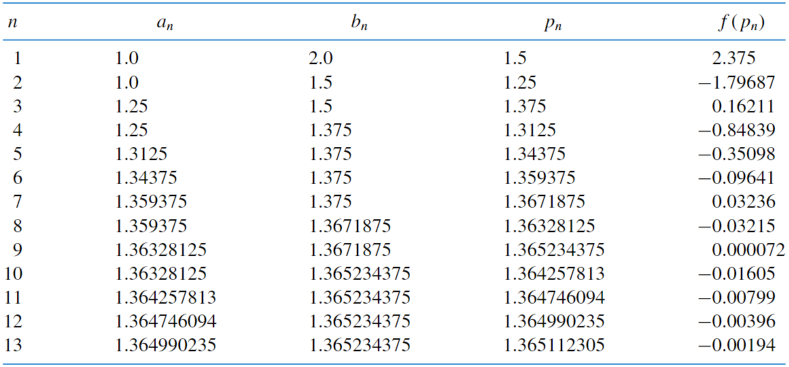
Now, $f(1.5) = 2.375 > 0$.

This indicates that we should select the interval $[1, 1.5]$ for our second iteration.

Then $f(1.25) = -1.796875$ so our new interval is $[1.25, 1.5]$, whose midpoint is 1.375.

Continuing in this manner gives the values in the following table:

The example, continued



n	a_n	b_n	p_n	$f(p_n)$
1	1.0	2.0	1.5	2.375
2	1.0	1.5	1.25	−1.79687
3	1.25	1.5	1.375	0.16211
4	1.25	1.375	1.3125	−0.84839
5	1.3125	1.375	1.34375	−0.35098
6	1.34375	1.375	1.359375	−0.09641
7	1.359375	1.375	1.3671875	0.03236
8	1.359375	1.3671875	1.36328125	−0.03215
9	1.36328125	1.3671875	1.365234375	0.000072
10	1.36328125	1.365234375	1.364257813	−0.01605
11	1.364257813	1.365234375	1.364746094	−0.00799
12	1.364746094	1.365234375	1.364990235	−0.00396
13	1.364990235	1.365234375	1.365112305	−0.00194

Final word on the bisection method

The Bisection method, though conceptually clear, has significant drawbacks.

It is relatively slow to converge (that is, N may become quite large before p_N is sufficiently close to a root), and a good intermediate approximation might be inadvertently discarded.

However, the method has the important property that it always converges to a solution.

For that reason, it is often used as a starter for the more efficient methods we will see later in this theme.