EM215

Numerical Methods

Prof. D.S.K. Karunasinghe,
Department of Engineering Mathematics,
Faculty of Engineering,
University of Peradeniya,
Sri Lanka.
February 2023.

Numerical Methods

Chapter 1: Introduction to numerical methods

- 1.1 Introduction
- 1.2 Taylor series
- 1.3 Error analysis
  - 1.3.1 Significant figures
  - 1.3.2 Accuracy and Precision
  - 1.3.3 Error Definitions
  - 1.3.4 Error as a stopping criteria
- 1.4 Round-off Errors
  - 1.4.1 Computer Representation of numbers
  - 1.4.2 Arithmetic manipulation of computer numbers
- 1.5 Truncation Errors
  - 1.5.1 Taylor series to estimate Truncation Errors
  - 1.5.2 Taylor series for numerical differentiation
  - 1.5.3 Error propagation
  - 1.5.4 Total numerical error

Chapter 2: Solutions to nonlinear equations

- 2.1 Root finding methods
- 2.2 Bracketing methods:
  - *Graphical methods, Bisection method*
- 2.3 Rate of convergence
- 2.4 Open methods:
  - *Fixed point iteration, Newton-Raphson method*
- 2.5 Systems of nonlinear equations

Chapter 3: Approximation and curve fitting

- 3.1 Least squares approximation
- 3.2 Fourier approximation

Reference:

Steven C. Chapra, Raymond P. Canale (2015). Numerical methods for engineers. 7th edition, McGraw-Hill publishers, New York.

Note: A soft copy of 7th edition of the above book is available on internet.

1.0 Introduction to numerical methods

1.1 Introduction

Imagine that you have to repair a car. Even if you have a very good tool-box, you cannot do the task if you do not understand how the car works. Also, the skill in using the appropriate tools is also needed.

Using computers to solve engineering problems also requires fundamental understanding of how engineering systems work and the skill to model and implement them for computer solution.

The next section introduces mathematical modeling in engineering problem solving, and uses a simple example to illustrate how numerical methods are involved in the problem solving process (Figure 1).

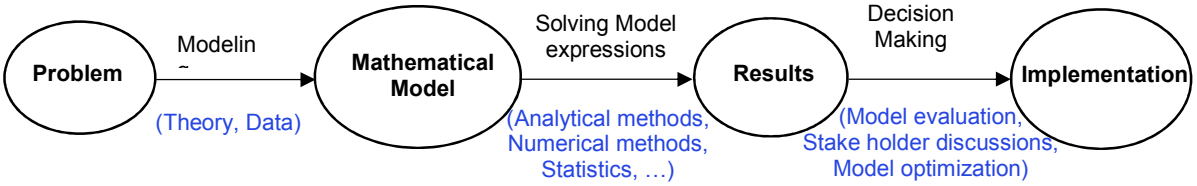


Figure 1: Engineering problem solving process

1.1.1 A Simple Mathematical Model

A mathematical model can be broadly defined as a formulation that expresses the essential features of a physical system or process in mathematical terms. A model can be represented by a functional relationship of the form (Chapra and Canale, 2015),

$$\begin{matrix} Dependent \\ variable \end{matrix} = f\left(\begin{matrix} independent \\ variables \end{matrix}, \quad parameters, \quad \begin{matrix} forcing \\ functions \end{matrix}\right).$$

- *Dependent variable* - a characteristic that usually reflects the behavior or state of the system
- *Independent variables* - dimensions, such as time and space, along which the system's behavior is being determined
- *Parameters* - constants reflective of the system's properties
- *Forcing functions* - external influences acting upon the system

The functional relationships of models can range from a simple algebraic relationship to large complicated sets of differential equations. These may be solved either using analytical methods or numerical methods.

1.1.2 A simple Example

Problem:

Consider a free-falling body of mass m near the earth's surface. Suppose it starts falling with an initial velocity of v_0 towards the earth. Determine its terminal velocity and how this velocity is reached over the time.

Modeling:

Use Newton's second law of motion $F = ma$.

Let m, v be the mass and the velocity of the body, and $F_G = mg$, F_A be the force due to gravity and the air resistance acting on the body. Here, g is the gravitational acceleration. If it is assumed that the drag force is proportional to velocity of the body, we get $F_A = -cv$ where c is the constant called the drag coefficient. If t is the time, these give, $a = \frac{F}{m} \rightarrow \frac{dv}{dt} = \frac{F_G - F_A}{m} = \frac{mg - cv}{m}$.

Mathematical model:

$$\frac{dv}{dt} = g - \frac{c}{m}v$$

Solving:

(1) Analytical solution

The terminal velocity is $\frac{mg}{c}$.

To determine how the velocity changes over time, we have to solve the above differential equation. It has an analytical solution.

$-\frac{m}{c} ln\left(g - \frac{c}{m}v\right) = t + K$ where K is a constant.

Assuming that at $t = 0$, $v = v_0$. Then,

$K = -\frac{m}{c} ln\left(g - \frac{c}{m}v_0\right)$, and $-\frac{m}{c} ln\left(g - \frac{c}{m}v\right) + \frac{m}{c} ln\left(g - \frac{c}{m}v_0\right) = t$

$\rightarrow \left(\frac{g - \frac{c}{m}v_0}{g - \frac{c}{m}v}\right)^{\frac{m}{c}} = e^t \rightarrow \left(g - \frac{c}{m}v_0\right)e^{-\frac{c}{m}t} = g - \frac{c}{m}v$

$$\boxed{v = \frac{mg}{c} - \frac{m}{c}\left(g - \frac{c}{m}v_0\right)e^{-\frac{c}{m}t}} \qquad t > 0$$

Now when we know m, c, g and v_0 we can determine the velocity over time.

For $m = 68.1\ kg$, $c = 12.5\ kg/s$, $g = 9.8\ m/s^2$ and $v_0 = 0$, the velocity change over time can be given as in Fig. 2.

(2) Numerical solution

The terminal velocity is $\frac{mg}{c}$.

The model expression $\frac{dv}{dt} = g - \frac{c}{m}v$ can be solved by approximating the derivative, $\frac{dv}{dt}$, when the velocity at a given time point is known.

Let us use $m = 68.1\ kg$, $c = 12.5\ kg/s$, $g = 9.8\ m/s^2$ and $v_0 = 0$ where velocity at time $t = 0$ is known.

For two points in time, $t_2 > t_1$, where $t_2 - t_1 = \Delta t$ is small, the derivative $\frac{dv}{dt}$ can be approximated as, $\frac{dv}{dt} \cong \frac{v_2 - v_1}{t_2 - t_1} = \frac{v_2 - v_1}{\Delta t}$, where v_1, v_2 are the velocities at times $t_1,\ t_2$.

$\therefore\ \frac{dv}{dt} \cong \frac{v_2 - v_1}{\Delta t} \cong g - \frac{c}{m} v_1$, which gives the approximation,

$$\boxed{v_2 = \left(g - \frac{c}{m} v_1\right) \Delta t + v_1}$$

We can iteratively solve the above expression for new values of v_2 starting from a known set of (t_1, v_1). Using $(t_0, v_0) = (0, 0)$ as starting(t_1, v_1), and a time interval of $\Delta t = 2\ s$,

$v_2 = \left(9.8 - \frac{12.5}{68.1} \times 0\right) \times 2 + 0 = 19.6\ m/s.$

Now take this newly found values as new (t_1, v_1). That is, $v_1 = 19.6\ m/s$ and $t_1 = t_{1(prev)} + \Delta t = 0 + 2 = 2\ s$, which give another set of v_2, t_2 as,

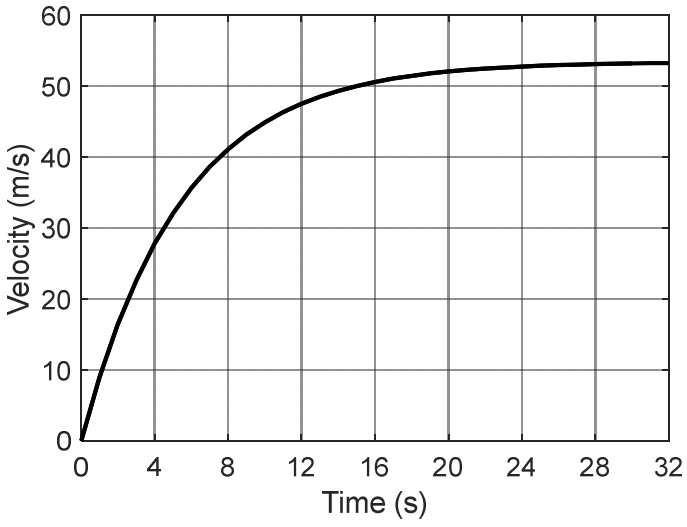
$v_2 = \left(9.8 - \frac{12.5}{68.1} \times 19.6\right) \times 2 + 19.6 = 32.00\ m/s$ and $t_2 = t_1 + \Delta t = 4\ s$.

Repeating the above procedure, the velocities over time can be determined. These numerical results with analytical results are shown in Fig. 3. The error calculated as,

$$error = true\ value - numerical\ approximation$$

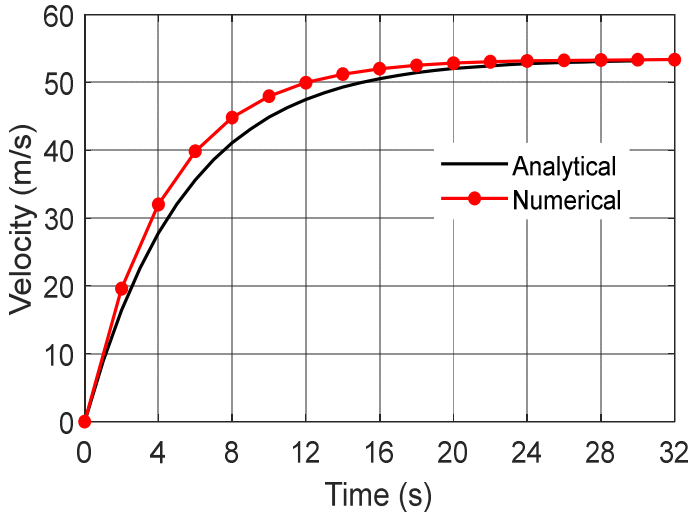
is shown in Fig. 4. Here, the analytical solution gives the true values.

Results:



Analytical solution	
Time (s)	Velocity (m/s)
0	0.00
1	8.95
2	16.41
3	22.61
4	27.77
6	35.64
8	41.10
10	44.87
12	47.49
16	50.56
32	53.24
∞	53.39

Figure 2: Graph and values produced by the analytical solution



Numerical Solution	
Time (s)	Velocity (m/s)
0	0.00
2	19.60
4	32.00
6	39.86
8	44.82
10	47.97
12	49.96
16	52.02
20	52.84
32	53.36
∞	53.39

Figure 3: Graphs of analytical and numerical solutions and the values produced by the numerical solution

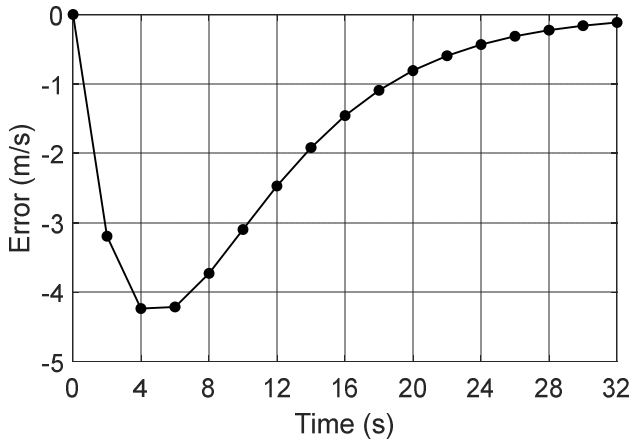


Figure 4: Error in the numerical solution

Ex 1.1:

Considering the example of a falling body, try to answer the following questions. Note that there can be more than one answer for each question.

(a) Explain how the step size (Δt) affect the accuracy of the numerical solution (velocity over time).

(b) Identify possible numerical/computational issues that can arise when using large/extremely small step sizes.

(c) How would determine a suitable step size for a given problem?

(d) Instead of the velocity at $t = 0$, if the velocity at $t = 10\ s$ was initially known as $v = 44.87\ m/s$, how would you numerically determine the velocities from $0 - 10\ s$?

Ex 1.2: 

In order to facilitate you understand the answers to the above exercise, use computer simulations. You may use Matlab or any other programming software you are comfortable with. Specifically,

(a) Use different Δt values (e.g. $\Delta t = 0.001, 0.1, 1, 5\ s$) and observe the errors in the solutions you get. (You may compare them with the analytical values).

(b) Use Δt values such as $10^{-500}, 10^{-100}, 10^{-10}, 10, 100\ s$ and investigate how they affect the solution and the computations.

(c) Implement your suggestions to part (d) in Ex. 1 and check their appropriateness.

1.2 Taylor series (*Covered in First year*)

Taylor series of a function of one variable

Taylor series of a function $f(x)$ about a points x^* is given by,

$$f(x) = f(x^*) + \frac{f'(x^*)}{1!}(x-x^*) + \frac{f''(x^*)}{2!}(x-x^*)^2 + \ldots + \frac{f^N(x^*)}{N!}(x-x^*)^N + \frac{f^{(N+1)}(\xi)}{(N+1)!}(x-x^*)^{N+1}.$$

The last term is called the remainder, and ξ is a real number between x and x^*.

Taylor series of a function of several variables

Consider a function of several variables $f(\boldsymbol{X})$ where, $\boldsymbol{X} = [x_1, x_2, \ldots, x_n]^T$ and $f: R^n \to R$.

Definition: *r^{th} Differential of f*

If all partial derivatives of the function f through order $r \geq 1$ exist and are continuous at a point $\boldsymbol{X}^*$, the polynomial

$$d^r f(\boldsymbol{X}^*) = \underbrace{\sum_{i=1}^{n}\sum_{j=1}^{n}\cdots\sum_{k=1}^{n}}_{r \text{ summations}} h_i h_j \ldots h_k \frac{\partial^r f(\boldsymbol{X}^*)}{\partial x_i \partial x_j \ldots \partial x_k}$$

is called the r^{th} differential of f at $\boldsymbol{X}^*$.

E.g. Expand $d^r f(\boldsymbol{X}^*)$ for $r = 2$ and $n = 3$.

$$d^2 f(\boldsymbol{X}^*) = d^2 f(x_1^*, x_2^*, x_3^*) = \sum_{i=1}^{3}\sum_{j=1}^{3} h_i h_j \frac{\partial^2 f(\boldsymbol{X}^*)}{\partial x_i \partial x_j}$$
$$= h_1^2 \frac{\partial^2 f(\boldsymbol{X}^*)}{\partial x_1^2} + h_2^2 \frac{\partial^2 f(\boldsymbol{X}^*)}{\partial x_2^2} + h_3^2 \frac{\partial^2 f(\boldsymbol{X}^*)}{\partial x_3^2}$$
$$+ 2h_1 h_2 \frac{\partial^2 f(\boldsymbol{X}^*)}{\partial x_1 \partial x_2} + 2h_1 h_3 \frac{\partial^2 f(\boldsymbol{X}^*)}{\partial x_1 \partial x_3} + 2h_2 h_3 \frac{\partial^2 f(\boldsymbol{X}^*)}{\partial x_2 \partial x_3}$$

The Taylor series expansion of a function $f(\boldsymbol{X})$ about a point $\boldsymbol{X}^*$ is

$$f(\boldsymbol{X}) = f(\boldsymbol{X}^*) + df(\boldsymbol{X}^*) + \frac{1}{2!}d^2 f(\boldsymbol{X}^*) + \frac{1}{3!}d^3 f(\boldsymbol{X}^*) + \ldots + \frac{1}{N!}d^N f(\boldsymbol{X}^*) + \frac{1}{(N+1)!}d^{N+1} f(\boldsymbol{X}^* + \theta \boldsymbol{h})$$

where the last term is called the remainder where $0 < \theta < 1$ and $\boldsymbol{h} = \boldsymbol{X} - \boldsymbol{X}^*$.

Revision exercise:

(a) Find the second-order Taylor series approximation of the function $f(x_1, x_2, x_3) = x_1^4 + x_2^2 x_3$ about the point $\boldsymbol{X}^* = [1,1,1]^T$.

(b) Find the second-order Taylor series approximation of the function $f(x_1, x_2, x_3) = e^{x_1} + x_2^2 x_3$ about the point $\boldsymbol{X}^* = [1,0,-2]^T$.

1.3 Error analysis

In the earlier example of a free-falling body, the availability of an analytical solution allowed us to compute the error exactly. For many applied engineering problems, we cannot obtain analytical solutions. Therefore, we cannot compute exactly the errors associated with our numerical methods. In such cases, we have to find approximations or estimates of the errors.

Errors (noise) can be introduced into a mathematical model by two possible sources: the measurement errors and model errors (see Fig. 1). Model errors can sometimes be due to idealization (or approximation) of the real situation. For example, in the falling-body example, we have assumed that the wind is absent and the air resistance to be constant. However, we do not discuss this type of errors (noise) in this course.

Numerical errors can be introduced when solving the models using computers. The two major forms of such numerical errors are (a) round-off errors and (b) truncation errors. *Round-off error* is due to the fact that computers can represent only quantities with a finite number of digits. *Truncation error* occurs when the numerical methods employ approximations to represent exact mathematical operations and quantities.

Apart from these, there can be other errors that are not directly connected with the numerical methods themselves, such as blunders, formulation errors etc.

The next few sections will introduce quantification of error. This will be followed by introductions to round off errors and truncation errors.

1.3.1 Significant figures

Numerical methods deal with approximations connected with the numbers.

Whenever we use a number in a computation, we must know how 'accurate' the number is. For example, in the falling-body problem, the mass was known up to 0.1 kg accuracy.

The significant figures, or digits, designate the reliability of numerical values. The significant digits of a number are those that can be used with confidence. They correspond to the number of 'certain digits' plus one 'estimated digit'. For example, the scale shown in Fig. 5(i), which measures some quantity up to < 10 units, reads 1.95, where 1.9 is the two certain digits and 0.05 is the estimated digit totaling to 3 significant digits (It is conventional to set the estimated digit at one-half of the smallest scale).

Zeros are not always significant figures because they may be necessary to locate a decimal point. For example, 0.0054, 0.054, 0.000054 all have two significant figures while the

number of significant figures in 5400 is not exactly clear. This confusion can be avoided by using scientific notation, where 5.4×10^2, 5.40×10^2 indicate 2, 3 significant figures respectively.

Numerical methods yield approximate results. One criteria to specify how confident we are in our approximate result is using significant figures. Also, the computers can retain only a finite number of significant figures. Therefore, when using quantities like, $e, \pi, \sqrt{2}$, which cannot be expressed exactly by a limited number digits, only an approximate value is represented by the computer. The omission of the remaining significant figures is called round-off error. Therefore, both the round-off error and the use of significant figures can indicate our confidence in a numerical result.

1.3.2 Accuracy and Precision

Accuracy refers to how closely a computed or a measured value agrees with the true value. Precision refers to how closely individual computed or measured values agree with each other. This can best be understood by an example of target practice. Figure 5 (ii) shows the hits by 4 gunners (bullseye is the target). On comparison, Fig. 5 (ii) shows that (a) is both precise and accurate, (b) is accurate but less precise, (c) is less accurate (biased) but precise, and (d) is both less accurate and less precise.

We prefer both precise and accurate results. In numerical methods, we use the term 'error' to represent both inaccuracy and imprecision.

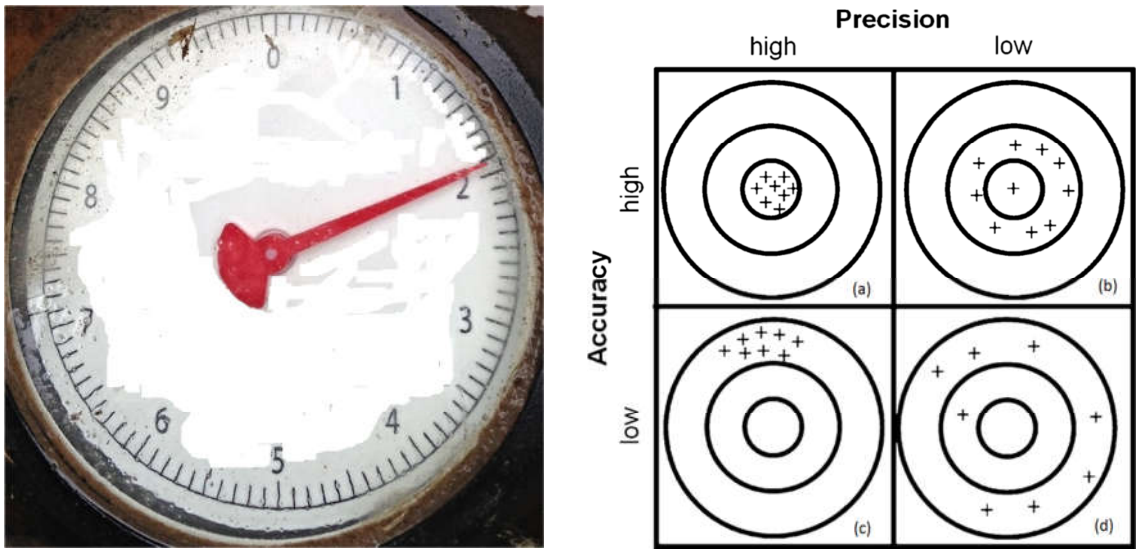


Figure 5. (i) Scale (ii) Hits by 4 gunners – categorized according to accuracy and precision

1.3.3 Error definitions

In general, error can be defined as,

$$True\ Error\ \ (E_t) = True\ value - Approximation.$$

In order to get an idea whether the error is large or small, we need a relative measure. The relative error is defined as,

$$Relative\ error = \frac{True\ Error}{True\ value}$$

and as a percentage as,

$$Percent\ Relative\ error\ (\varepsilon_t) = \frac{True\ Error}{True\ value} \times 100\%.$$

However, in real-world situations, we do not know the $True\ value$ or the answer, *a priori*. Therefore, error has to be approximated using the best estimate of the true value as follows.

$$Approximate\ Error\ (E_a) = Best\ estimate\ of\ True\ value - Approximation$$

$$\varepsilon_a = \frac{Approximate\ error}{Best\ estimate\ of\ True\ value} \times 100\%$$

When numerical methods use iterative methods with successive approximations (e.g. in calculating e by successively adding terms, in the falling-object problem, successive approximations to reach the terminal velocity), the percent relative error can be determined as,

$$\varepsilon_a = \frac{Current\ approximation - previous\ approximation}{Current\ approximation} \times 100\%.$$

Ex: In falling-body object problem, when successively approximating velocities to find the terminal velocity, how do you know when you have reached the terminal velocity (assume that you do not know the value of terminal velocity)?

Note: Note that the error can be negative or positive. In most of the applications, we are more interested in the magnitude of the error rather than whether it is positive or negative. Therefore, the absolute values of the above quantities are often used.

1.3.4 Error as a stopping criteria

When performing successive approximations, we need to determine when we are have to stop calculations. The errors E_a and ε_a can both be useful as stopping criteria.

* Iterative calculations may be stopped when the absolute error is smaller a pre-defined tolerance

$$|\varepsilon_a| < \varepsilon_0 \qquad \text{or} \qquad |E_a| < E_0.$$

We can relate these errors to the number of significant figures in the approximation using the result of Scarborough (1966). To assure that a result is correct to at least n significant figures the following criteria has to be met.

$$\boxed{\varepsilon_0 = (0.5 \times 10^{2-n})\%}$$

* Instead, if errors keep on increasing, *i.e.*,

$$|E_{a,i}| < |E_{a,i+1}| < |E_{a,i+2}| < \cdots \qquad \text{or} \qquad |\varepsilon_{a,i}| < |\varepsilon_{a,i+1}| < |\varepsilon_{a,i+2}| < \cdots$$

where $|E_{a,k}|, |\varepsilon_{a,k}|$ are the errors at k^{th} iteration, it may be indicative that the approximations are not converging and the calculation has to be stopped. The numerical method/inputs may have to be changed, to achieve convergence.

Ex 1.3

Using the infinite series, $e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \ldots$, and an appropriate stopping criteria, estimate $e^{0.8}$ correct to 3 significant figures. Calculate the absolute approximate error and the absolute percentage relative error in each successive approximation.

1.4 Round-off Errors

Round-off errors originate from the fact that computers can retain only a fixed number of significant figures during storing and in calculations. Therefore, apart from irrational numbers like $\pi, e, \sqrt{2}$, etc., certain rational numbers also cannot be represented exactly by a computer.

1.4.1 Computer representation of numbers

Round-off errors arise due to the way the numbers are stored in computers. Information is represented as *words* that consists of *bits*.

Integer representation

One approach for this is 'signed magnitude method' where the first bit indicate the sign (0 for positive and 1 for negative), and the rest store the number in binary form. Since zero does not need a sign, the negative representation of zero is used to represent one more negative number.

e.g. (1) The integer -100 in 16-bit computer will be stored as in Fig. 6.

(2) Negative representation of zero of 16-bit computer can be used to store -2^{15} as in Fig. 7.

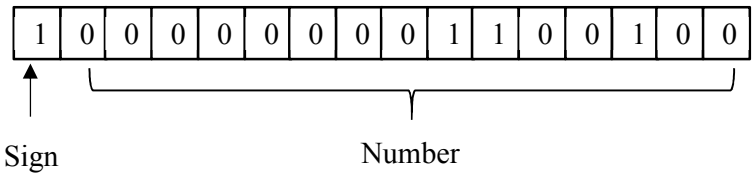


Figure 6. Representation of -100 in a 16-bit computer.

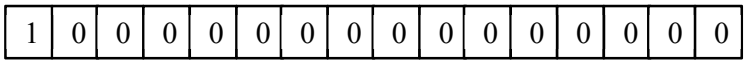


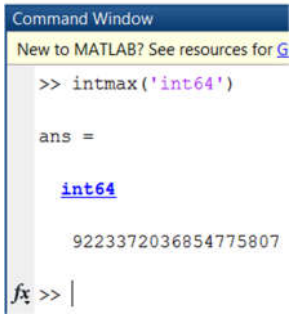
Figure 7. Representation of $-2^{15} = -32768$ in a 16-bit computer.

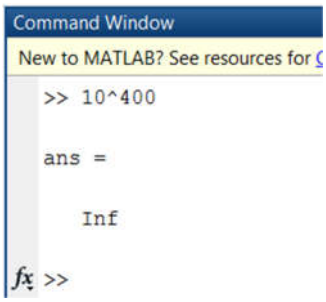
Ex: What are the largest and the smallest integers that a 64-bit computing machine can store with the signed magnitude method?

The example above serves to show that computers are limited in storing numbers; i.e. they cannot represent integers larger than a certain integer and integers smaller than a certain integer. Same (and even worse) is true with real numbers as we see next.

Ex: Find the largest integer for different integer types defined in Matlab.

Type 10^400 in Matlab prompt and Enter. What is the result you get?





Floating point representation

In this approach, a number is expressed as a fractional part, called a mantissa or significand, and an integer part, called an exponent or characteristic, as in the following expression and in Fig. 8.

$$\boxed{mb^e}$$

m – mantissa (significand)
b – base of the number
e – exponent (characteristic)

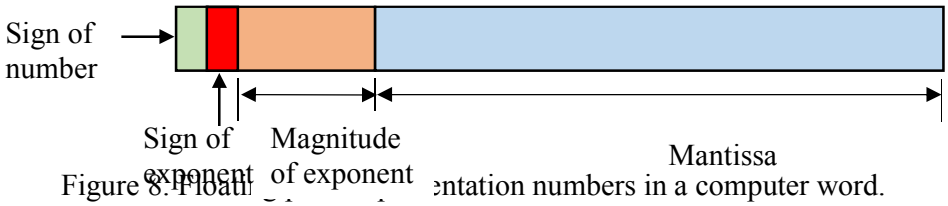


Figure 8. Floating point representation numbers in a computer word.

Mantissa holds the fractional component and the exponent holds an integer. An example of such a representation using 7-bit word is shown in Fig. 9. If the mantissa has leading zero digits (e.g. 0.012345, 0.0033333'), it is normalized such that,

$$\frac{1}{b} \leq m < 1$$

allowing more significant digits to be stored. Thus for base 10, $0.1 \leq m < 1$and for base 2, $0.5 \leq m < 1$.

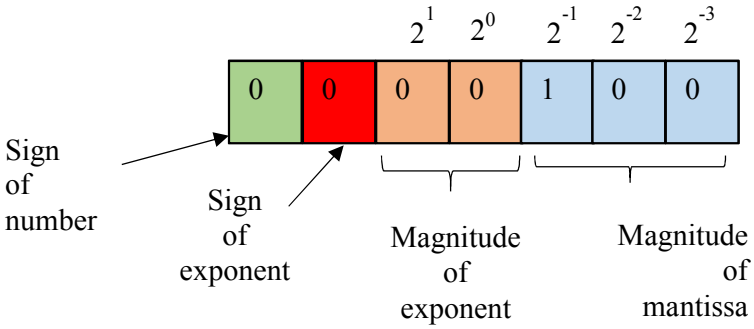


Figure 9. Floating point representation using a 7-bit word.

Floating- point representation allows very large/small numbers be stored. However, that too is limited because both the exponent and the mantissa can hold finite numbers of figures.

Ex 1.4:

List all the real numbers that can be stored using 7-bit word shown in Fig. 9 (Give the values using base 2 and 10).

(a) Are the numbers equally spaced?
(b) Comment on the distance between adjacent numbers starting from zero to the largest.
(c) What are the largest and smallest positive values that this word can store?

In general, is the number of real numbers that can be stored using a finite number of bits, finite or infinite?

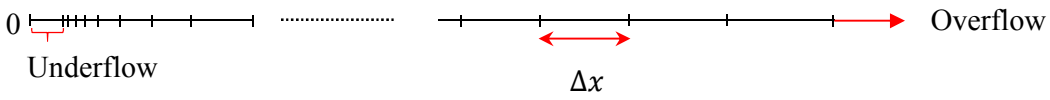


Figure 10. Floating point representation in computers.

The interval between adjacent numbers (Δx, see Fig. 10) can be expressed as,

$$\frac{|\Delta x|}{|x|} \leq \varepsilon \text{ for chopping,}$$
$$\frac{|\Delta x|}{|x|} \leq \frac{\varepsilon}{2} \text{ for rounding,}$$

where ε referred to as **machine epsilon** is defined as,

$$\varepsilon = b^{1-t}$$

where b is the base and t is the number of significant digits in the mantissa.

The errors introduced by approximating (either by chopping or rounding) a given number is called quantizing error.

The round off errors arising in computer representation of numbers are, however, negligible for most of engineering applications. For cases where more precision is required, extended precision is available with commonly used software (e.g. Matlab, Excel). They use double precision which can represent numbers from approximately$10^{-308} - 10^{308}$.

1.4.2 Arithmetic manipulations of computer numbers

Normalized base-10 numbers are used here to illustrate the effect of round-off errors on arithmetic operators: simple addition, subtraction, multiplication, and division. A hypothetical decimal computer with a 4-digit mantissa and a 1-digit exponent, with chopping, is used to illustrate the effects of the operations.

Operation	Example
Addition: - The smaller exponent is modified so that the exponents are same. - Two numbers are then added and the result is chopped.	Numbers to be added: $0.1234 \times 10^1, \quad 0.5678 \times 10^{-1}$ $0.1234 \times 10^1, \quad 0.005678 \times 10^1$ 0.1234×10^1 0.005678×10^1 0.1290×10^1
Subtraction: - Identical to addition, except the two numbers are subtracted *Three non-significant zeros are appended to the answer. Any subsequent calculations will consider these zeros as significant. Loss of significance during the subtraction of nearly equal numbers is one of the greatest source of round-off error in numerical methods.	Subtract 5.677 from 5.678: 0.5678×10^1 -0.5677×10^1 0.0001×10^1 The answer is represented as, 0.1000×10^{-3}.
Multiplication:	Numbers to be multiplied: $0.1234 \times 10^3, \quad 0.5678 \times 10^{-1}$

- The exponents are added and the mantissas are multiplied. The answer is normalized and chopped.	$0.1234 \times 10^3 \times 0.5678 \times 10^{-1}$ $= 0.07006652 \times 10^2$ $= 0.7006 \times 10^1$
Division: - Performed similar to multiplication.	

Round-off errors on large numbers of interdependent computations

Certain problems/methods require extremely large numbers of interdependent arithmetic manipulations to arrive at their final results (e.g. Finite Element Methods, Numerical optimization). Although the individual round-off error could be small, when accumulated over a large number of computations, it can become significant.

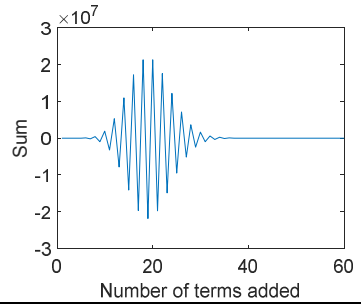
e.g. Quantizing errors can be accumulated over repeated summations	`format long`
	`A=10^-5;`
	`N=1000000;`
	`sumA=0;`
	`for k=1:N`
	`  sumA=sumA+A;`
	`end`
	`sumA`
	`sumA =`
	`9.999999999790591`

e.g. Adding a large number to a small number This can happen in computation of infinite series. *Will computer show that* $\sum_{n=1}^{\infty} \frac{1}{n}$ *divergent???*	Using a hypothetical decimal computer with 4-digit mantissa and 1-digit exponent, add 5000 and 0.01. 0.5000×10^4 0.000001×10^4 0.500001×10^4 When chopped, $= 0.5000 \times 10^4$
Subtractive cancellation: Round-off error is introduced when subtracting two nearly equal floating-point numbers.	

Smearing:
Smearing occurs whenever the individual terms in a summation are larger than the summation itself.
e.g. Calculating e^{-20} using the infinite series.
	`%-------------- exp(-20)-----------`
	`x=-20;`
	`sume=1;`
	`sumA=[];  % summation values`
	`for k=1:125`
	`  sume=sume+(x^k)/factorial(k);`
	`  sumA=[sumA;sume];`
	`end`
	`plot(sumA)`
	`sume  % final answer`
	`sume =`

ex: Try $(e^{-1})^{20}$.

```
3.431944085424785e-05
```



Computing in extended precision is the general strategy to reduce round off errors. Although there are rules of thumb to mitigate round-off error, trial and error may be needed to actually determine effects of the errors on a computation. Taylor series, will provide a mathematical approach for estimating these effects.

1.5 Truncation Errors

Truncation errors are those that result from using an approximation in place of an exact mathematical procedure.

e.g. We approximated the acceleration of the falling body (page 3) as , $\frac{dv}{dt} \cong \frac{\Delta v}{\Delta t} = \frac{v_2 - v_1}{t_2 - t_1}$.

Taylor series (page 82 of the text book) is widely used to express functions in an approximate from.

Considering two points x_i, x_{i+1}, which are close to each other, if the function f and its first $n+1$ derivatives are continuous on an interval containing x_i and x_{i+1}, the Taylor series expansion can be given as,

$$f(x_{i+1}) = f(x_i) + \frac{f'(x_i)}{1!}(x_{i+1} - x_i) + \frac{f''(x_i)}{2!}(x_{i+1} - x_i)^2 + \cdots + \frac{f^n(x_i)}{n!}(x_{i+1} - x_i)^n + R_n.$$

The remainder, R_n, accounts for all terms from $n+1$ to infinity, and it can be expressed as,

$$R_n = \frac{f^{n+1}(\xi)}{(n+1)!}(x_{i+1} - x_i)^{n+1}.$$

For a step size $h = x_{i+1} - x_i$, these become,

$$\boxed{f(x_{i+1}) = f(x_i) + \frac{f'(x_i)}{1!}h + \frac{f''(x_i)}{2!}h^2 + \cdots + \frac{f^n(x_i)}{n!}h^n + R_n}$$

and (1)

$$\boxed{R_n = \frac{f^{n+1}(\xi)}{(n+1)!}h^{n+1}}$$

where ξ is a point that lies between x_i and x_{i+1} (page 87 of the text book). (2)

Taylor series can be used to approximate a function by an n^{th} order polynomial taking the summation of the first n terms. The truncation error in this approximation is R_n, and is also expressed as,

$$O(h^{n+1})$$

which means that the error is of the order of h^{n+1}.

Problem: Consider the function, $f(x) = \sin x$. Based on the point $x = \pi/6$, use Taylor series approximations from $n = 0$ to 4 to estimate $\sin x$ at $x = \pi/4$.

Solution: Let's take $x_i = \pi/6$ and $x_{i+1} = \pi/4$.

Therefore, $h = x_{i+1} - x_i = \pi/4 - \pi/6 = \pi/12$.

The zero-order approximation $(n = 0)$ is,

$f(x_{i+1}) \cong f(x_i) \rightarrow f(\pi/4) \cong f(\pi/6) \rightarrow \sin\pi/4 \cong \sin\pi/6 = 0.5000.$

This gives a percent relative error of $\varepsilon_t = \left|\frac{0.7071-0.5000}{0.7071}\right| \times 100 = 29.29\%$

The first-order approximation $(n = 1)$ is,

$f(x_{i+1}) \cong f(x_i) + \frac{f'(x_i)}{1!}(x_{i+1} - x_i)$

$\rightarrow f(\pi/4) \cong \sin\pi/6 + \frac{\cos\pi/6}{1!}(\pi/12) = 0.7267.$

This gives a percent relative error of $\varepsilon_t = \left|\frac{0.7071-0.7267}{0.7071}\right| \times 100 = 2.77\%$

The second-order approximation $(n = 2)$ is,

$f(x_{i+1}) \cong f(x_i) + \frac{f'(x_i)}{1!}(x_{i+1} - x_i) + \frac{f''(x_i)}{2!}(x_{i+1} - x_i)^2$

$\rightarrow f(\pi/4) \cong \sin\pi/6 + \frac{\cos\pi/6}{1!}(\pi/12) - \frac{\sin\pi/6}{2!}(\pi/12)^2 = 0.7096.$

This gives a percent relative error of $\varepsilon_t = \left|\frac{0.7071-0.7096}{0.7071}\right| \times 100 = 0.35\%$

Similarly third-order approximation $(n = 3)$ give,

$f(\pi/4) \cong 0.7070,$ and $\varepsilon_t = 0.01\%.$

Fourth-order approximation $(n = 4)$ give,

$f(\pi/4) \cong 0.7071,$ and $\varepsilon_t = 0\%.$

1.5.1 Taylor series to estimate Truncation Errors

In the falling-body problem we wanted to predict velocity as a function of time $(v(t))$. If we apply Taylor series (let t_i, t_{i+1} be two points on the time axis),

$$v(t_{i+1}) = v(t_i) + \frac{v'(t_i)}{1!}(t_{i+1} - t_i) + \frac{v''(t_i)}{2!}(t_{i+1} - t_i)^2 + \cdots + \frac{v^n(t_i)}{n!}(t_{i+1} - t_i)^n + R_n.$$

If we truncate the series after first derivative,

$$v(t_{i+1}) = v(t_i) + \frac{v'(t_i)}{1!}(t_{i+1} - t_i) + R_1.$$

Solving for $v'(t_i)$,

$$v'(t_i) = \frac{v(t_{i+1}) - v(t_i)}{(t_{i+1} - t_i)} + \frac{R_1}{(t_{i+1} - t_i)}.$$

First order approximation Truncation error

Truncation error in approximating the derivative is,

$$R_1' = \frac{R_1}{(t_{i+1} - t_i)} = \frac{v''(\xi)}{2!}(t_{i+1} - t_i),$$

$$R_1' = O(t_{i+1} - t_i).$$

1.5.2 Taylor series for numerical differentiation

Finite divided differences

Forward difference approximation of the first derivative

Consider the following forward expansion of the Taylor series,

$$f(x_{i+1}) = f(x_i) + \frac{f'(x_i)}{1!}h + \frac{f''(x_i)}{2!}h^2 + \cdots$$

Truncating this after the first derivative and re-arranging,

$$\boxed{f'(x_i) \cong \frac{f(x_{i+1}) - f(x_i)}{h}}$$

This was the one we used in falling-body problem.

Backward difference approximation of the first derivative

Consider the following backward expansion of the Taylor series,

$$f(x_{i-1}) = f(x_i) - \frac{f'(x_i)}{1!}h + \frac{f''(x_i)}{2!}h^2 - \cdots$$

Truncating this after the first derivative and re-arranging, (3)

$$\boxed{f'(x_i) \cong \frac{f(x_i) - f(x_{i-1})}{h}}.$$

You may have used this in your first lab assignment problem.

Centered difference approximation of the first derivative

Subtracting the backward expansion of the Taylor series from the forward expansion,

$f(x_{i+1}) = f(x_{i-1}) + \frac{2f'(x_i)}{1!}h + \frac{2f^3(x_i)}{3!}h^3 + \cdots$

Solving the above for $f'(x_i)$

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_{i-1})}{2h} - \frac{f^3(x_i)}{3!}h^2 - \cdots$$

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_{i-1})}{2h} - O(h^2).$$

The centered difference approximation,

$$\boxed{f'(x_i) \cong \frac{f(x_{i+1}) - f(x_{i-1})}{2h}}$$

therefore, has a truncation error of the order h^2.

Approximation of higher order derivatives

Higher order differences can also be derived from the Taylor series. For example, adding the forward and backward expansion of the Taylor series gives,

$$f(x_{i+1}) - 2f(x_i) + f(x_{i-1}) = \frac{2f''(x_i)}{2!}h^2 + \frac{2f^4(x_i)}{2!}h^4 + \cdots$$

This can give the following *centered* approximation for the second order derivative,

$$f''(x_i) \cong \frac{f(x_{i+1}) - 2f(x_i) + f(x_{i-1})}{h^2}$$

With the order of its error being $O(h^2)$.

1.5.3 Error propagation

Here we are going to study how the error in numbers propagate through mathematical functions. For example, if some number x is represented by $\tilde{x}$ with some error, we may want to estimate how much error will be there in $e^{\tilde{x}}$ evaluated with this number. In general we want to estimate,

$$\Delta f(\tilde{x}) = |f(x) - f(\tilde{x})|.$$

Note that, however, both x and $f(x)$ are unknown.

If $f(x)$ is continuously differentiable and if $\tilde{x}$ is sufficiently close to x, we can use Taylor series as,

$$f(x) = f(\tilde{x}) + \frac{f'(\tilde{x})}{1!}(x - \tilde{x}) + \frac{f''(x_i)}{2!}(x - \tilde{x})^2 + \cdots.$$

Using only the first order approximation,

$$f(x) - f(\tilde{x}) \cong f'(\tilde{x})(x - \tilde{x})$$

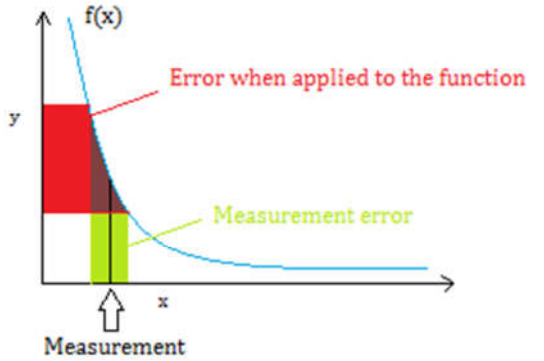
which gives

$$\boxed{\Delta f(\tilde{x}) \cong |f'(\tilde{x})(x - \tilde{x})| = |f'(\tilde{x})|\Delta\tilde{x}; \qquad \Delta\tilde{x} = |(x - \tilde{x})|}.$$

Therefore, we can get an estimate for the error of the function if we have estimates for the derivative and the error of the variable.

Using the Taylor series expansion for functions of several variables, and ignoring the second and higher order terms, the above result can be extended for functions of several variables as,

$$\boxed{\Delta f(\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_n) \cong \left|\frac{\partial f}{\partial x_1}\right| \Delta\tilde{x}_1 + \left|\frac{\partial f}{\partial x_2}\right| \Delta\tilde{x}_2 + \ \ldots + \left|\frac{\partial f}{\partial x_n}\right| \Delta\tilde{x}_n}.$$



Ex 1.5: The deflection y of a beam is given by,

$$y = \frac{FL^4}{8EI}$$

where F is the loading (N/m), L is the length (m), E is the modulus of elasticity (N/m^2), and I is the moment of inertia (m^4).

(a) Estimate the error in y given the following data.

$\tilde{F} = 750$ N/m; $\tilde{L} = 5$ m; $\tilde{E} = 7.5 \times 10^9$ N/m^2; $\tilde{I} = 0.0005$ m^4;

$\Delta\tilde{L} = 0.02$ m; and $\Delta\tilde{F},\ \Delta\tilde{E},\ \Delta\tilde{I}$ are negligible.

(b) Estimate the error in y given the following data.

$\tilde{F} = 750$ N/m; $\tilde{L} = 5$ m; $\tilde{E} = 7.5 \times 10^9$ N/m^2; $\tilde{I} = 0.0005$ m^4;
$\Delta\tilde{F} = 30$ N/m; $\Delta\tilde{L} = 0.02$ m; $\Delta\tilde{E} = 5 \times 10^7$ N/m^2; $\Delta\tilde{I} = 0.000005$ m^4.

Stability and condition

Condition of a problem indicates its sensitivity to changes in its input values. A computation is said to be **numerically unstable** if the uncertainty of the input values is magnified by the numerical method.

The relative error of $f(x)$ can be approximated as (using the earlier result),

$$\frac{f(x) - f(\tilde{x})}{f(x)} = \frac{\Delta f(\tilde{x})}{f(x)} \cong \frac{f'(\tilde{x})(x - \tilde{x})}{f(\tilde{x})}.$$

The relative error of x can be estimated as,

$$\frac{x - \tilde{x}}{x} \cong \frac{x - \tilde{x}}{\tilde{x}}.$$

The **condition number** is defined as the ratio of the relative errors as,

$$condition\ number = \frac{f'(\tilde{x})\tilde{x}}{f(\tilde{x})}.$$

Thus a condition number larger than 1 in magnitude indicates that the error in x is magnified by $f(x)$, and vice versa. Functions with very large condition numbers are said to be *ill-conditioned*.

Ex 1.7:

(a) Compute and interpret the condition number for,
$f(x) = \cos x$ for $\tilde{x} = \pi/2 + 0.1(\pi/2)$
and
$f(x) = \cos x$ for $\tilde{x} = \pi/2 + 0.01(\pi/2)$.

(b) Compute the condition number for the function $f(x) = x^3$.

(c) Comment on the relative error of the function $f(x) = e^x$

(d) Comment on the condition number of the function $f(x) = e^x$

1.5.4 Total numerical error

The total numerical error is the summation of the truncation and round-off errors. Generally, as the truncation errors decrease the round off errors increase (see Fig. 11). The optimal step size for a given computation would be the one that minimizes the sum of the two errors. In practical applications, however, the round-off error is not a serious issue as most computers carry enough significant figures.

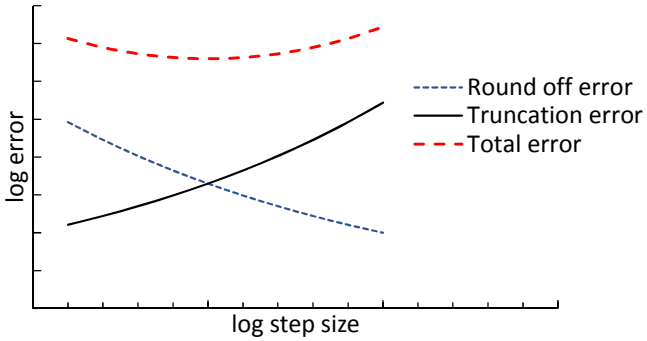


Figure 11. Round off and Truncation errors with step size.

Ex 1.8: Estimate the first derivative of the following function at $x = 0.5$ with a step size of 1 and centered difference approximation. Then divide the step size by 10 and calculate the derivative. Repeat the calculation to observe how the round-off error grows.

$$f(x) = -0.1x^4 - 0.5x^2 - 0.5x + 1.2$$

Note that $f'(x) = -0.4x^3 - 1.0x - 0.5$ and $f'(0.5) = -1.050$.

Centered difference approximation of the derivative is $f'(x_i) \cong \frac{f(x_{i+1}) - f(x_{i-1})}{2h}$, and $x_i = 0.5$.

h	x_{i+1}	x_{i-1}	$f(x_{i+1})$	$f(x_{i-1})$	$f'(x_i)$	$\lvert E_t \rvert$
1.000000000	1.500000000	-0.500000000	-1.1812500000000	1.3187500000000	-1.2500000000000	0.2000000000000
0.100000000	0.600000000	0.400000000	0.7070400000000	0.9174400000000	-1.0520000000000	0.0020000000000
0.010000000	0.510000000	0.490000000	0.8081847990000	0.8291851990000	-1.0500200000000	0.0000200000000
0.001000000	0.501000000	0.499000000	0.8176993497999	0.8197993501999	-1.0500002000000	0.0000002000000
0.000100000	0.500100000	0.499900000	0.8186449934998	0.8188549935002	-1.0500000020003	0.0000000020003
0.000010000	0.500010000	0.499990000	0.8187394999350	0.8187604999350	-1.0500000000191	0.0000000000191
0.000001000	0.500001000	0.499999000	0.8187489499994	0.8187510499994	-1.0500000000024	0.0000000000024
0.000000100	0.500000100	0.499999900	0.8187498950000	0.8187501050000	-1.0499999997249	0.0000000002751
0.000000010	0.500000010	0.499999990	0.8187499895000	0.8187500105000	-1.0499999980595	0.0000000019405
0.000000001	0.500000001	0.499999999	0.8187499989500	0.8187500010500	-1.0499999758551	0.0000000241449

Ex 1.9: 

You may use either Python or Matlab to solve the followings.

(a) Solve the Ex. 8 using the forward difference approximation of the first derivative, and produce the results as a table (or matrix).
(b) Solve the Ex. 8 using the centered difference approximation of the second derivative, and produce the results as a table (or matrix).

To Read:

Read Section 4.3.2 and Section 4.4 of the text book yourself.

By now, you are expected to have learnt the followings:

- Understanding the distinction between accuracy and precision.
- Learning how to quantify error.
- Learning how error estimates can be used to decide when to terminate an iterative calculation.
- Understanding how round-off errors occur because digital computers have a limited ability to represent numbers.
- Understanding why floating-point numbers have limits on their range and precision.
- Recognizing that truncation errors occur when exact mathematical formulations are represented by approximations.
- Knowing how to use the Taylor series to estimate truncation errors.
- Understanding how to write forward, backward, and centered finite-difference approximations of the first and second derivatives.
- Recognizing that efforts to minimize truncation errors can sometimes increase round-off errors.

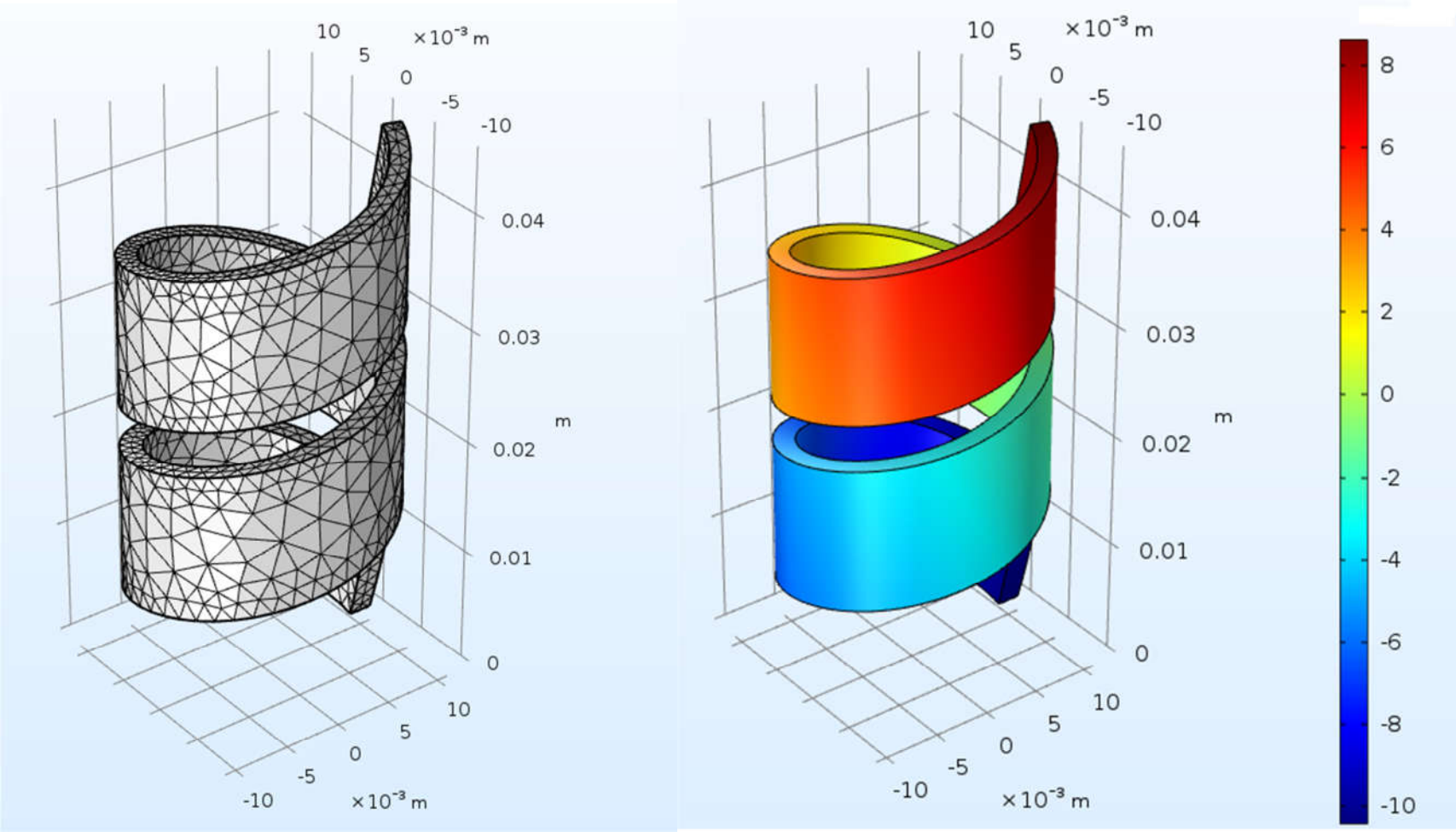
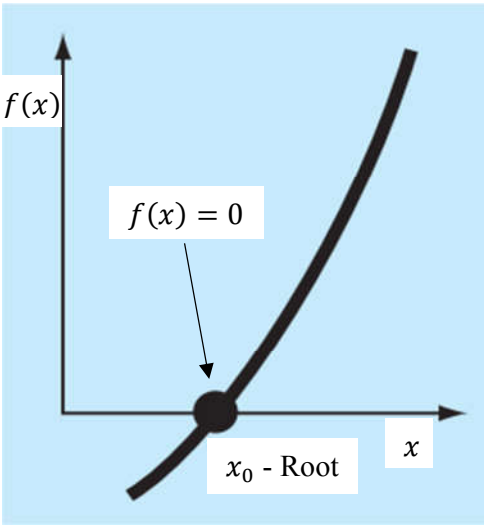


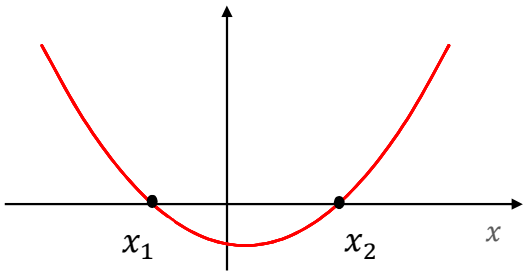
Figure 12. An example of large number of interdependent computations: Finite element analysis of a machine component.

2.0 Solutions to nonlinear equations



(a)

$$f(x) = ax^2 + bx + c$$



$$f(x) = ax^2 + bx + c = 0$$

$$\rightarrow x_1,\quad x_2 = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

(b)

Figure 2.1 (a) Root (b) Roots of a quadratic equation

A **root** of an equation is a value of x that makes $f(x) = 0$. Roots are also called **zeros** of function $f(x)$.

Although indirectly, roots of equations frequently occur in engineering design, where model equations involving dependent and independent variables are used (e.g. Heat balance, mass balance, etc.). Finding extreme values (maxima, minima) of a function also includes root finding, where we solve $f'(x) = 0$.

e.g. The model equation derived for the velocity of the falling body.

$$\boxed{v = \frac{mg}{c} - \frac{m}{c}\left(g - \frac{c}{m}v_0\right)e^{-\frac{c}{m}t}} \qquad t > 0$$

Here, the velocity, v is *explicitly* expressed as a function of time.

Suppose our problem was to find the drag coefficient, c that will enable the body to attain a given velocity within a given time. In that case, there is no way we can isolate c on one side of the equal sign and c is said to be *implicit*.

Ex 2.1: Consider the falling body problem. Find the value of the drag coefficient, c that will make a body of mass $m = 68.1$ kg attains a velocity of $v = 44.87$ m/s within a time $t = 10$ s. Take $g = 9.8$ m/s and $v_0 = 0$. Note that we know the answer from our previous examples as, $c = 12.5$ kg/s. How can you find this answer with the above model and the given values?

Hint: Note that we can define a function, $f(c) = v - \frac{mg}{c} + \frac{m}{c}\left(g - \frac{c}{m}v_0\right)e^{-\frac{c}{m}t}$, and the answer we expect is a root of the equation $f(c) = 0$.

2.1 Root finding methods

Definitions:

Algebraic functions: A function $y = f(x)$ is algebraic if it can be expressed in the form,

$$f_n y^n + f_{n-1} y^{n-1} + \cdots + f_1 y + f_0 = 0$$

where f_i is an i^{th} order polynomial in x.

[Alternative definition: An algebraic function is a function $f(x)$ which satisfies $p\big(x, f(x)\big) = 0$, where $p(x, y)$ is a polynomial in x and y with integer coefficients.]

Polynomials: Polynomials are a simple of class of algebraic functions that can be written as

$$f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$$

where n is the order of the polynomials and a_i values are constants.

Transcendental functions: Functions which are not algebraic are called transcendental functions. These include trigonometric, exponential, logarithmic etc. functions.

Algebraic equation: statement of the equality of two expressions formulated by applying to a set of variables the algebraic operations, namely, addition, subtraction, multiplication, division, raising to a power, and extraction of a root.

Root finding methods fall into two problem areas:

1. The determination of real roots of algebraic and transcendental equations.

   These methods use the prior knowledge of the approximate location of a root to find it. Bracketing methods (bisection, false position) and open methods (fixed point iteration, Newton-Raphson) used for this purpose will be discussed in this course.

2. The determination of all real and complex roots of polynomials. This includes more systematic methods.

   Matlab function, ***roots*** gives roots of polynomials.

2.2 Bracketing methods

These methods are called bracketing methods because they use two initial guesses ($x_l,\ x_u$) lying either side of the root (Fig. 2.2). Methods then try to reduce the width of the bracket iteratively. Repeated application of these methods always results in closer estimates of the true value of the root, and these methods are said to be convergent.

2.2.1 Graphical methods

Graphical method is extremely useful to get a rough estimate for a root. These can serve as initial guesses for the other methods.

Ex 2.2:

(a) Solve Ex. 2.1 graphically.

(b) Find the minimum of the function.

$$f(x) = 0.65 - \frac{0.75}{1+x^2} - 0.65x \tan^{-1}\frac{1}{x}, \qquad x > 0$$

Graphical methods are also useful for understanding the properties of the functions and identifying possible problematic areas when using the numerical methods. For example, identifying areas where the graph is flat, changes rapidly, may contain multiple roots, etc.

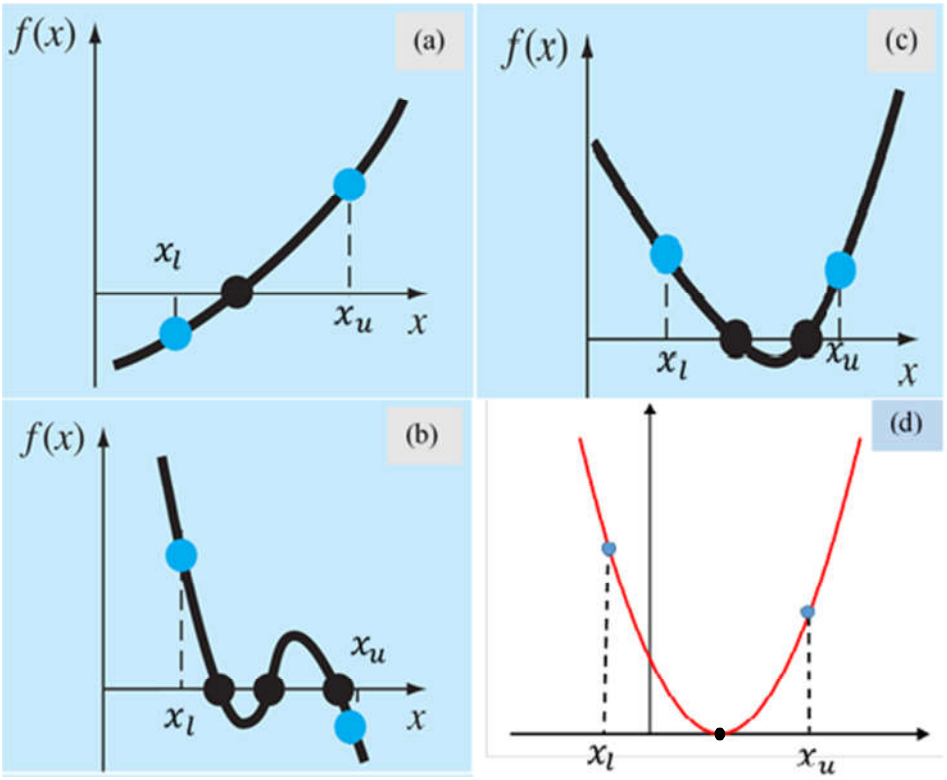


Figure 2.2 (a), (b): x_l, x_u are of opposite signs and contain at least one root. (c), (d): x_l, x_u are of same sign and contain several roots. Note that (d) has a multiple root.

Ex 2.3:

Use computer graphics to locate roots of equation,

$$f(x) = \sin 10x + \cos 3x$$

over the range $[0, 5]$. Are there any multiple roots?

2.2.2 Bisection method (interval halving method, Bolzano's method)

When $f(x)$ changes sign on opposite sides of the root, bisection method can be used. If $f(x)$ is real and continuous in an interval $[x_l, x_u]$ and if $f(x_l)f(x_u) < 0$, then there is at least one real root in between x_l and x_u.

Algorithm:

Step 1: Choose initial guesses x_l and x_u such that $f(x_l)f(x_u) < 0$.

Step 2: Obtain an estimate of the root x_r as,

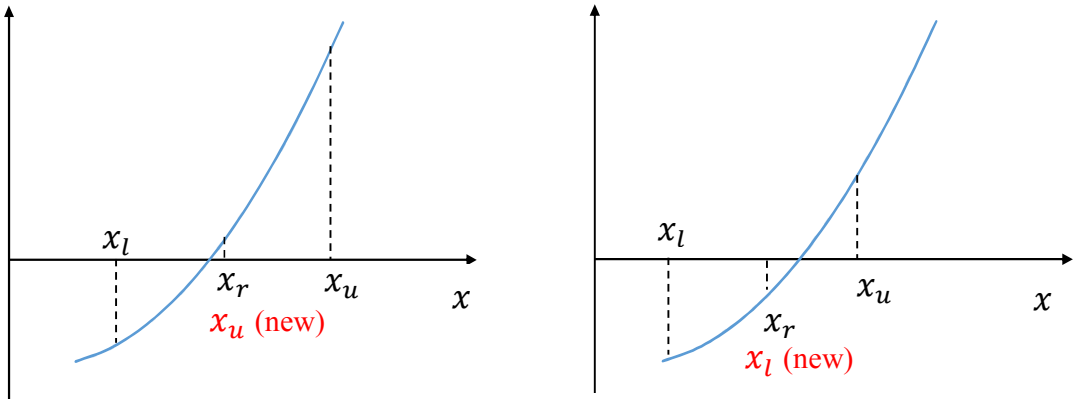
$$x_r = \frac{x_l + x_u}{2}$$

Step 3: Make the following evaluations to determine in which subinterval the root lies:

(a) If $f(x_l)f(x_r) = 0$, terminate the computation.

(b) If $f(x_l)f(x_r) > 0$, set $x_l = x_r$ and return to step 2.

(c) If $f(x_l)f(x_r) < 0$, set $x_u = x_r$ and return to step 2.



(i) (ii)

Figure. 2.3. Assigning the mid-point, x_r: (i) the step 3(c) (ii) the step 3(b)

Ex 2.4: Use bisection method to estimate the root between $x = 3.5$ and 4.0 of the function $f(x) = \sin 10x + \cos 3x$ so that the absolute error does not exceed

(a) 0.05, (b) 0.005.

Stopping criteria:

A practical stopping criteria is to determine whether the estimated root is sufficiently close to the true root. Note that the true error is (x^* is the actual root),

$$|x^* - x_r| \leq \frac{x_u - x_l}{2}.$$

Therefore, the percent relative error can be approximated as,

$$\varepsilon_a = \left(\frac{x_u - x_l}{2}\right) \Big/ \left(\frac{x_u + x_l}{2}\right) \times 100 = \left|\frac{x_u - x_l}{x_u + x_l}\right| \times 100\%.$$

Ex 2.4: Repeat the above exercise with $x_l = 3.7$ and $x_u = 4.7$ so that the percent relative error does not exceed 0.1%. (Ans: 3.74296875).

Number of iterations to reach a desired absolute error bound E_d:

Absolute error of the initial guesses, $E_0 \leq x_u^0 - x_l^0$
Absolute error of the values in the i^{th} iteration, $E_i \leq x_u^i - x_l^i$
Considering the possible maximum values of these absolute errors in each iteration, it can be seen that,

$$E_0 = 2E_1 = 2^2E_2 = \cdots = 2^nE_n$$

If we need that $E_n \leq E_d$,

$$\frac{E_0}{2^n} \leq E_d$$

$$2^n \geq \frac{E_0}{E_d}$$

$$n \log 2 \geq \log\left(\frac{E_0}{E_d}\right)$$

$$\boxed{n \geq \log\left(\frac{E_0}{E_d}\right)/\log 2 = \log_2\left(\frac{E_0}{E_d}\right)}$$

Ex 2.5: Check the above result with the solutions of Ex 2.4.

2.3 Rate of convergence

Definition: If a sequence $x_1, x_2, \ldots, x_n$ converges to a value x^* and if there exist real numbers $\lambda > 0$ and $\alpha \geq 0$ such that

$$\lim_{n\to\infty} \frac{|x_{n+1} - x^*|}{|x_n - x^*|^\alpha} = \lambda$$

then α is called the rate of convergence of the sequence.
When $\alpha = 1$ we say the sequence converges *linearly* and when $\alpha = 2$ we say the sequence converges *quadratically*. If $1 < \alpha < 2$ then the sequence is said to exhibit *super-linear* convergence.

Rate of convergence of the bisection method

Let us denote the estimate of the root, x_r at the i^{th} iteration as x_r^i.
From our earlier derivations, $|x_r^n - x^*| \cong E_n/2$.
This gives,

$$\frac{|x_r^{n+1} - x^*|}{|x_r^n - x^*|} \cong \frac{E_{n+1}}{E_n} = \frac{E_0}{2^{n+1}}\frac{2^n}{E_0} = \frac{1}{2}.$$

Therefore, the bisection method converges linearly.

Ex 2.6: Plot the approximate error in Ex 2.4 to see if it shows linear behaviour. Comment on the gradient of the line. Note that you have to plot E_{n+1} vs E_n.

2.4 Open methods

Open methods use formulas to predict the root using only a single starting value or two starting values not necessarily bracket the root. The open methods, therefore, sometimes diverge.

2.4.1 Fixed point iteration

Definition: A fixed point of a function $f(x)$ is a point x_0 such that

$$f(x_0) = x_0.$$

Fixed point iteration method

A nonlinear equation of the form $f(x) = 0$ can be rewritten as,

$$f(x) = g(x) - x = 0$$
$$\rightarrow \boxed{g(x) = x}.$$

Therefore, the fixed point(s) of $g(x)$ is a root(s) of $f(x) = 0$.

Often, there are many ways to convert $f(x) = 0$ to one of $g(x) = x$ form. The simplest is $g(x) = x + \phi(x)f(x)$ for any function $\phi(x)$. However, it is important to find a $g(x)$ for which fixed point iteration converges.

Algorithm: Starting with an initial guess x_0 perform the following recursive process.

$$\boxed{x_{n+1} = g(x_n), \quad n = 0, 1, 2, \ldots}$$

If x_n converges to some x^*, then x^* is a fixed point of $g(x)$ and it is a solution of $f(x) = 0$. If the sequence converges, when n is large, x_n can be considered an approximate solution for $f(x) = 0$.

Stopping criteria: Percent relative error, defined as follows comparing two adjacent estimates, can be used as a stopping criteria for fixed point iteration.

$$\varepsilon_a = \left|\frac{x_{i+1} - x_i}{x_{i+1}}\right| \times 100\%.$$

Theorem 1: Let $g(x)$ be a continuous function on the interval $[a, b]$. If $g(x) \in [a, b]$, for each $x \in [a, b]$, then $g(x)$ has a fixed point in $[a, b]$. Furthermore, if $g(x)$ is differentiable on (a, b) and there exists a constant k such that

$$|g'(x)| \leq k < 1, \quad x \in (a, b),$$

then $g(x)$ has exactly one fixed point in $[a, b]$.

Theorem 2 (Fixed point theorem): Let $g(x)$ be a continuous function on the interval $[a, b]$. If $g(x) \in [a, b]$, for each $x \in [a, b]$, and if there exists a constant k such that

$$|g'(x)| \leq k < 1, \quad x \in (a, b),$$

then the sequence of iterates $\{x_k\}_{k=0}^{\infty}$ converges to a unique fixed point x^* of $g(x)$ in $[a, b]$, for any initial guess $x_0 \in [a, b]$.

Convergence

When x^* is a fixed point of $g(x)$, the fixed point iteration converges,

linearly when $g'(x^*) \neq 0$ *(see page 150 of the text book)*

and quadratically when $g'(x^*) = 0$ but $g''(x^*) \neq 0$.

The above convergence results can be shown using Taylor series as well.

Ex. 2.7: (a) Compute a fixed point of $g(x) = \cos x$ in the interval $[0, 1]$.

(b) Use the following formulations of fixed point iteration to find the root of the equation $x^3 - 7x + 2 = 0$ in $[0, 1]$, and explain why they converge/diverge.

(i) $g(x) = x^3 - 6x + 2 = x$, with $x_0 = 0.5$.

(ii) $g(x) = \frac{x^3+2}{7} = x$, with $x_0 = 0.5$.

(iii) $g(x) = \frac{x^3+2}{7} = x$, with $x_0 = 5.0$.

Ex. 2.8: Find c in Ex. 2.1 using fixed point iteration, with percent relative error < 0.1%.

The two-curve graphical method

Suppose we have to find the roots of $f(x) = 4\log x - x = 0$. We can separate the equation into two parts,

$$f_1(x) = x$$
$$f_2(x) = 4\log x$$

Then the x values corresponding to $f_1(x) = f_2(x)$ or the intersection of the two curves give the roots of $f(x) = 0$ as shown in Fig. 2.4

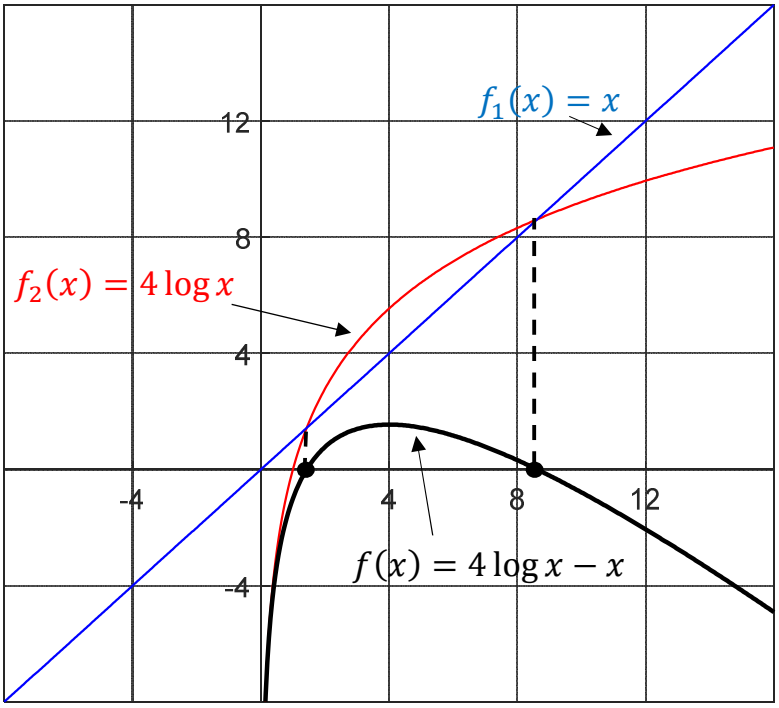


Figure 2.4 Intersection of two curves giving the roots

Figure 2.4 shows that the fixed point iteration $x_{n+1} = 4\log x_n$, diverges for starting values less than x_{r1}, and converges to x_{r2} for starting values greater than x_{r1} where $x_{r1} < x_{r2}$ are the two roots of $f(x) = 0$.

In general, the ways fixed point iterations can converge/diverge, monotonically/oscillatory are shown in Fig. 2.5.

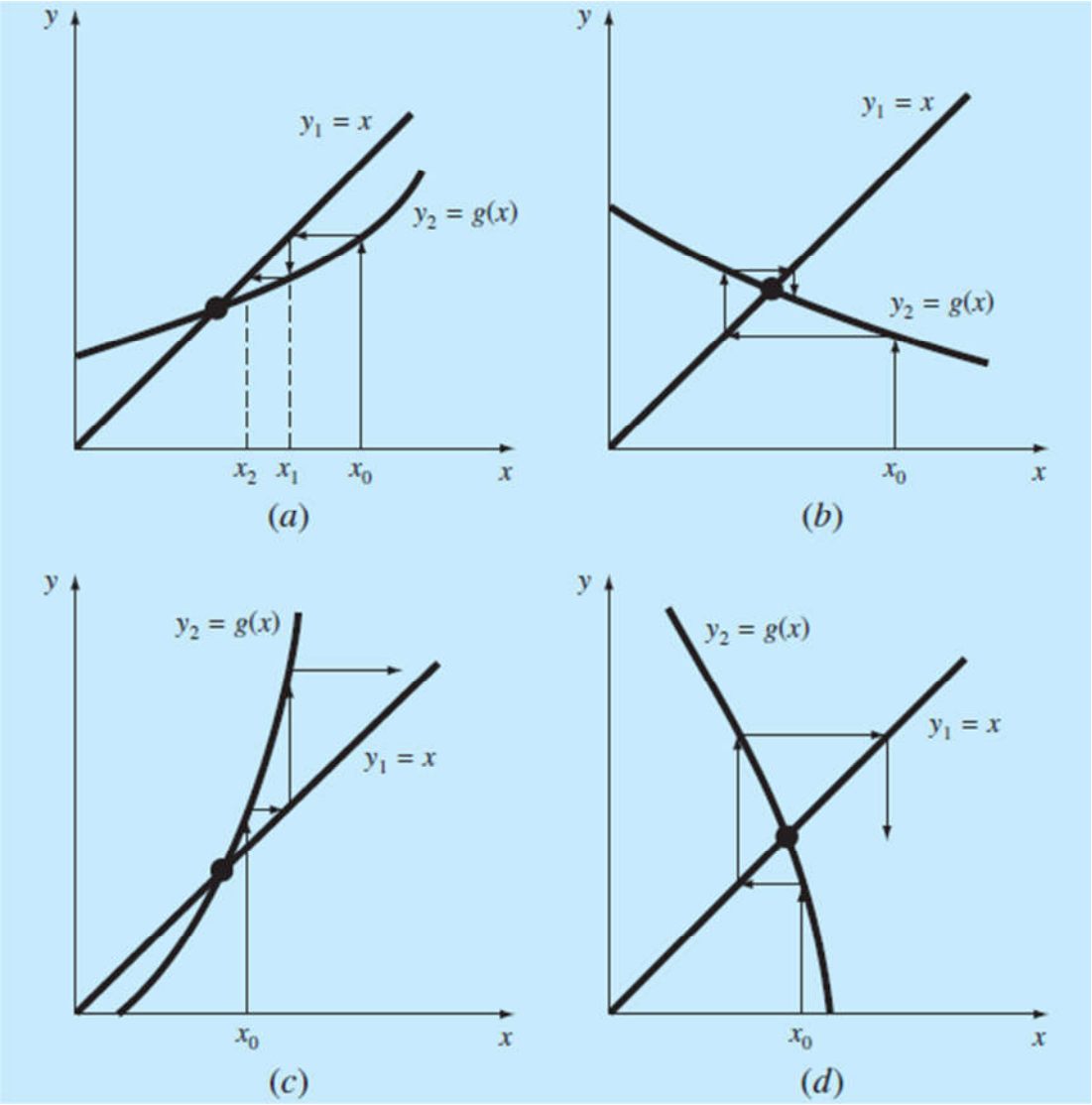


Figure 2.5 Convergence and divergence of fixed point iteration method (a) monotonic convergence (b) oscillatory convergence (c) monotonic divergence (d) oscillatory divergence.

2.4.2 Newton-Raphson method

This is perhaps the most widely used root finding method. Using the geometric interpretation illustrated in Fig. 2.6 the Newton-Raphson method can be derived as follows.

Starting with a point x_n, the next point, x_{n+1} is the point that the tangent at $\big(x_n, f(x_n)\big)$ cuts the x axis. This gives,

$$f'(x_n) = f(x_n)/(x_n - x_{n+1}) \quad \rightarrow \quad x_n - x_{n+1} = f(x_n)/f'(x_n)$$

which can be re-arranged as,

$$\boxed{x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \qquad n = 0, 1, 2, \ldots}.$$

This is called the Newton-Raphson formula.

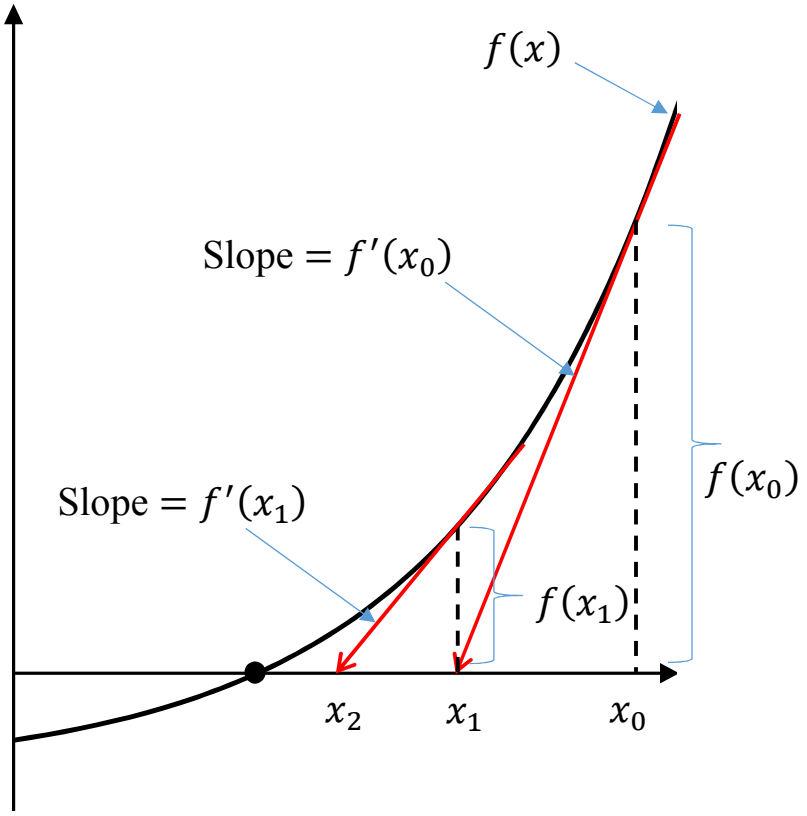


Figure 2.6 How N-R method works

Ex. 2.9: (a) Let $f(x) = x^2 - 2$. Find the positive root of $f(x) = 0$.
(b) Estimate the root of $f(x) = e^{-x} - 1$ with an initial guess of $x_0 = 1$.

Stopping criteria: As in fixed point iteration, percent relative error can be used.

Error analysis:
Let x^* be the root. Consider the Taylor series expansions,

$$f(x_{n+1}) = f(x_n) + \frac{f'(x_n)}{1!}(x_{n+1} - x_n) + \frac{f''(\xi)}{2!}(x_{n+1} - x_n)^2,$$

When x_{n+1} is closer to root, $x_{n+1} \cong x^*$ and $f(x_{n+1}) \cong 0$. These give,

$$0 \cong f(x_n) + \frac{f'(x_n)}{1!}(x^* - x_n) + \frac{f''(\xi)}{2!}(x^* - x_n)^2.$$

Using the N-R formula,

$$f(x_n) = -f'(x_n)(x_{n+1} - x_n).$$

Taking the sum of the above two expressions,

$$0 \cong f'(x_n)(x^* - x_{n+1}) + \frac{f''(\xi)}{2!}(x^* - x_n)^2$$

$$\rightarrow 0 \cong f'(x_n)E_{n+1} + \frac{f''(\xi)}{2!}{E_n}^2$$

When converging, both x_n and ξ gets closer to the root, x^* and the above can be approximated as,

$$\boxed{E_{n+1} \cong -\frac{f''(x^*)}{2f'(x^*)} {E_n}^2}.$$

This shows that the N-R method converges quadratically when closer to the root.

Ex. 2.10: Verify the above result with the solutions you got in Ex. 2.9.

Some points to be considered with Newton – Raphson method

- Check for $f'(x_n)$. Close to zero values can cause errors.
- Check for $f(x_n) = 0$ upon convergence to verify that it is a root.
- Check for possible divergence.
- Multiple roots.
- Initial guess must be sufficiently close to the true root.
- Some problematic cases are illustrated in Fig. 2.7.

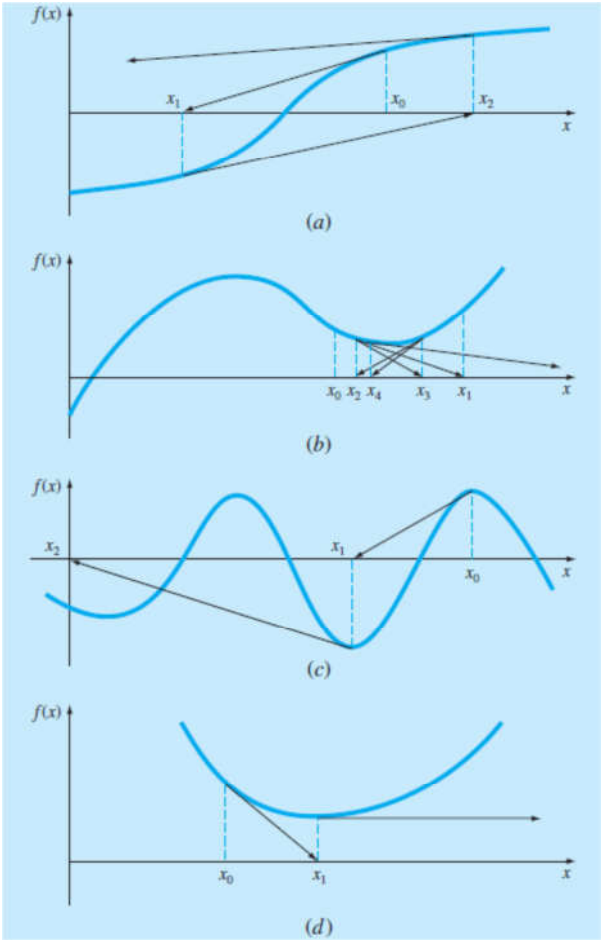


Figure 2.7 Some cases where N-R shows poor convergence

2.5 Systems of nonlinear equations

Consider a system of nonlinear equations of n variables, $x_1, x_2, \dots, x_n$.

$$f_1(x_1, x_2, \dots, x_n) = 0$$
$$f_2(x_1, x_2, \dots, x_n) = 0$$
$$\vdots$$
$$f_n(x_1, x_2, \dots, x_n) = 0$$

Solution is the set of x values, $(x_1^*,\ x_2^*, \dots,\ x_n^*)$ which solves the above equations.

Example: Solve the following system of nonlinear equations.

$$x^2 + xy = 10$$
$$y + 3xy^2 = 57$$

We may write these as,

$$u(x, y) = x^2 + xy - 10 = 0$$
$$v(x, y) = y + 3xy^2 - 57 = 0$$

The N-R method can be extended for functions of several variables. For two variables, the resulting expressions can be given as (see pages 171-172),

$$x_{n+1} = x_n - \frac{u_n \frac{\partial v_n}{\partial y} - v_n \frac{\partial u_n}{\partial y}}{\frac{\partial u_n}{\partial x}\frac{\partial v_n}{\partial y} - \frac{\partial u_n}{\partial y}\frac{\partial v_n}{\partial x}}$$

$$y_{n+1} = y_n - \frac{v_n \frac{\partial u_n}{\partial x} - u_n \frac{\partial v_n}{\partial x}}{\frac{\partial u_n}{\partial x}\frac{\partial v_n}{\partial y} - \frac{\partial u_n}{\partial y}\frac{\partial v_n}{\partial x}}$$

$$J = \frac{\partial u_n}{\partial x}\frac{\partial v_n}{\partial y} - \frac{\partial u_n}{\partial y}\frac{\partial v_n}{\partial x}$$

is called the Jacobian of the system.

We have

$$\frac{\partial u}{\partial x} = 2x + y; \qquad \frac{\partial u}{\partial y} = x; \qquad \frac{\partial v}{\partial x} = 3y^2; \qquad \frac{\partial v}{\partial y} = 1 + 6xy$$

Using initial values $x_0 = 1.5$ and $y_0 = 3.5$,

$$u_0 = -2.5; \quad v_0 = 1.625; \quad \frac{\partial u_0}{\partial x} = 6.5; \quad \frac{\partial u_0}{\partial y} = 1.5; \quad \frac{\partial v_0}{\partial x} = 36.75; \quad \frac{\partial v_0}{\partial y} = 32.5;$$

$J = 156.125$.

$$x_1 = 1.5 - \frac{-2.5\times32.5 - 1.625\times1.5}{156.125} = 2.0360 \qquad y_1 = 3.5 - \frac{1.625\times6.5 + 2.5\times36.75}{156.125} = 2.8439$$

Iterations can continue ….

True root is $(2,\ 3)$.

You are expected to have learnt the followings in Chapter 2:

1. Understand the graphical interpretation of a root
2. Know the bisection method and its convergence
3. Understand the difference between bracketing and open methods for root location
4. Understand the concepts of convergence and divergence; use the two-curve graphical method to provide a visual manifestation of the concepts
5. Know why bracketing methods always converge, whereas open methods may sometimes diverge
6. Realize that convergence of open methods is more likely if the initial guess is close to the true root
7. Understand the concepts of linear and quadratic convergence and their implications for the efficiencies of the fixed-point-iteration and Newton-Raphson methods
8. Understand the problems posed by multiple roots
9. Know the possibility to extend the Newton-Raphson approach to solve systems of nonlinear equations