ENGG 407
Numerical Methods in Engineering

Lecture #01

Dr. Sameh Nassar

L02 W12

Practical Information

L02 Instructor:

Dr. Sameh Nassar, P.Eng.

snassar@ucalgary.ca **CCIT 361F** **(403) 210-7822**

- **B.Sc.** (1995), Civil Engineering.
- **M.Sc.** (1999), Civil/Geomatics Engineering.
- **Ph.D.** (2004), Geomatics Engineering (U of C).

© Sameh Nassar

Practical Information

Office Hours:

- Open door policy

  (a request should be made in advance by email)

- Email questions are preferred

  (answer will be given within 48 hours)

© Sameh Nassar

Practical Information

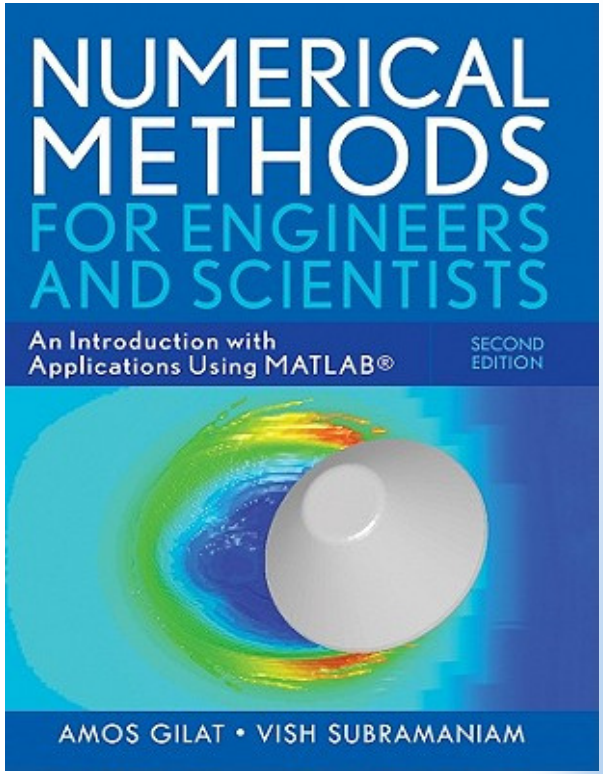
Required Textbook:

"**Numerical Methods for Engineers and Scientists**"

2nd Edition, 2011

Amos Gilat & Vish Subramaniam

John Wiley & Sons, Inc.



© Sameh Nassar

Practical Information

Course Grading:

Assignments (*5 or 6*): 30%

Quiz #1 (*Feb 15*): 15%

Quiz #2 (*Mar 15*): 15%

Final Exam: 40%

© Sameh Nassar

Practical Information

Teaching Assistants:

T02:

Herve Lahamy
(403) 220-4113
ENE 227F
hdalaham@ucalgary.ca

T04:

Essam Hamza
(403) 220-4370
ENE 227A
ehhamza@ucalgary.ca

© Sameh Nassar

Introduction to Numerical Methods

Remember from Previous Physics, Mechanics or Math Courses:

- In practice, a mathematical problem is derived from a physical phenomenon with making some simplifying assumptions to allow mathematical representation.

- However, the mathematical problem does not solve the physical problem exactly!

© Sameh Nassar

Numerical Methods: Definitions

- "**Numerical Methods** are techniques by which mathematical problems are formulated so that they can be solved with arithmetic operations".

- "**Numerical Methods** are techniques for approximating, in an efficient manner, the solutions to mathematically expressed problems, where the efficiency of the method depends on the accuracy required & the ease of implementation".

- "**Numerical Methods** use ***numbers*** to simulate mathematical processes, which in turn usually simulate real-world situations".

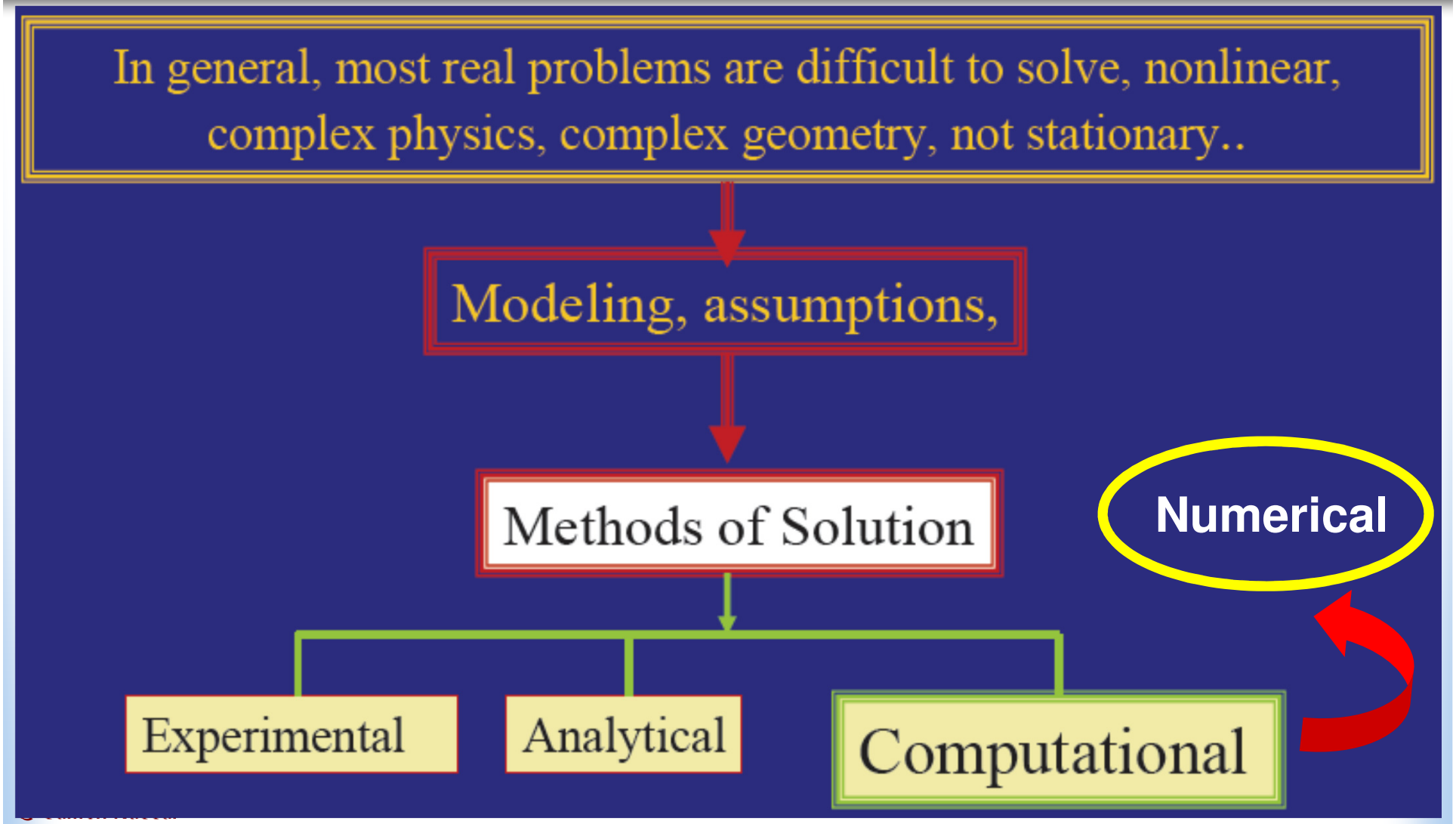
© Sameh Nassar

Numerical Methods: Strategy

- In general, the mathematical problem does not solve the physical problem exactly. Thus, it is more appropriate to find an approximation solution to a more complicated mathematical model.
- To obtain such an approximation, a method "or an algorithm" is developed.
- The algorithm consists of a sequence of algebraic and logical operations that produces the approximation to the mathematical problem (with a hope to the physical problem too!) within a prescribed accuracy.

© Sameh Nassar

Numerical Methods



Numerical Methods & Engineers

- **Civil Engineering**
  - Structure health monitoring.....
- **Electrical Engineering**
  - Design of electric circuits.....
- **Biomedical & Biomechanical Engineering**
  - Modeling of motion of joints.....
- **Mechanical Engineering**
  - Vibration analysis.....
- **Chemical Engineering**
  - Gas Laws, Reactors.....
- **Geomatics Engineering**
  - Satellite tracking, Position determination.....

© Sameh Nassar

ENGG 407 Topics

1. Introduction and Mathematical Background (*Ch. #1, #2*)
2. "*Roots of*" Nonlinear Equations (*Ch. #3*)
3. Linear Equations and Systems (*Ch. #4*)
4. Interpolation, Least-squares Estimation and Curve Fitting (*Ch. #5*)
5. Numerical Differentiation (*Ch. #6*)
6. Numerical Integration (*Ch. #7*)
7. Ordinary Differential Equations: Initial Value Problems (*Ch. #8*)
8. Ordinary Differential Equations: Boundary Value Problems (*Ch. #9*)
9. Optimization (*Tentative*).

© Sameh Nassar

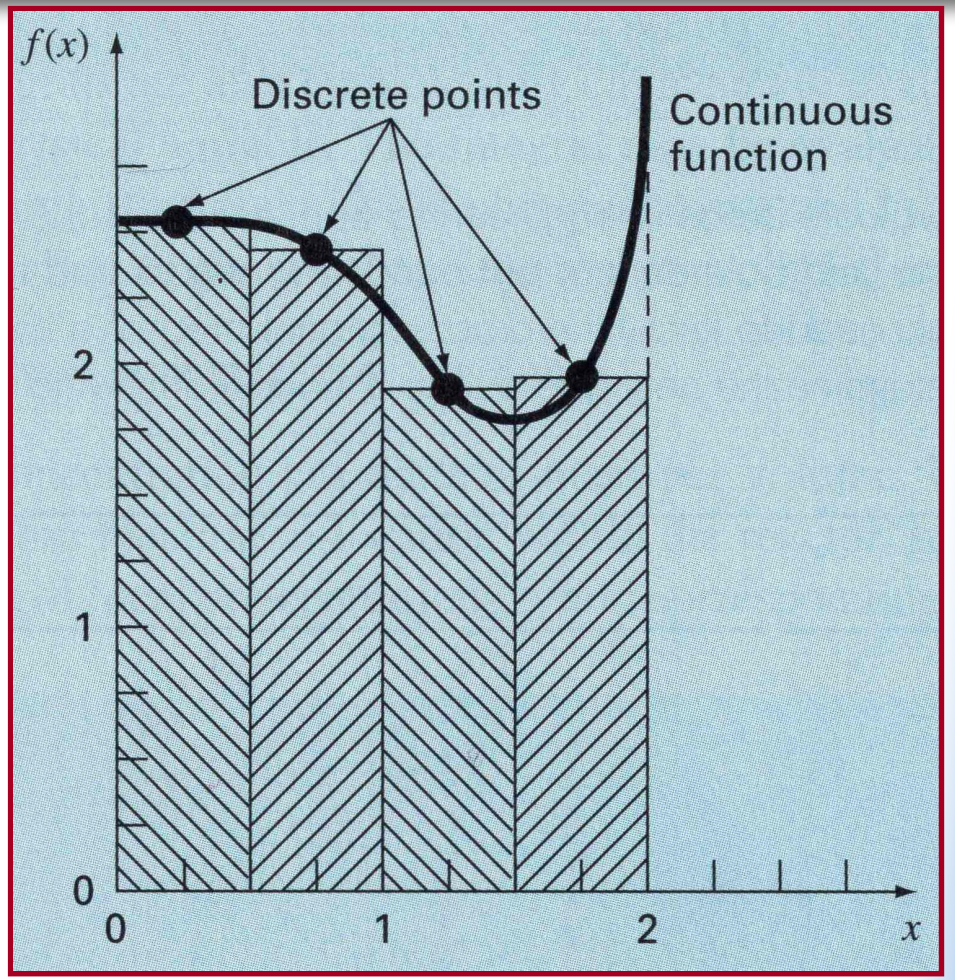
Why Numerical Methods? *"Case #I"*

Complicated continuous function

$$\int_0^2 \frac{2 + \cos\left(1 + x^{3/2}\right)}{\sqrt{1 + 0.5 \sin x}} \, e^{0.5x} \, dx$$

x	$f(x)$
0.25	2.599
0.75	2.414
1.25	1.945
1.75	1.993

Tabulated function



Numerical integration

© Sameh Nassar

Why Numerical Methods? *"Case #I"*

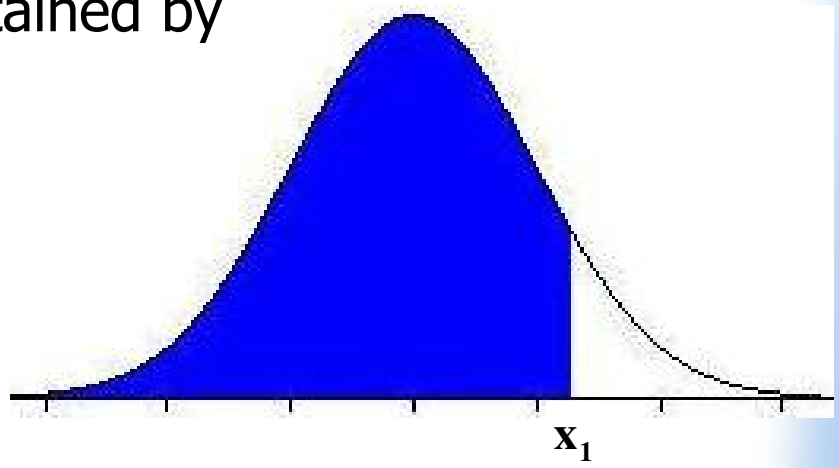
Recall from ENGG 319 (Probability & Statistics for Engineers):

- The density of the normal random variable X, with mean μ and variance σ^2, is:

$$n(x;\mu,\sigma) = f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\left(\frac{1}{2\sigma^2}\right)(x-\mu)^2}$$

- The probability for a specific value $\mathbf{x_1}$ (area under the curve from $-\infty$ to $\mathbf{x_1}$) is obtained by integrating the above equation. i.e.:

$$P(x < x_1) = \int_{-\infty}^{x_1} \frac{1}{\sqrt{2\pi}\sigma} e^{-\left(\frac{1}{2\sigma^2}\right)(x-\mu)^2} dx$$



© Sameh Nassar

Why Numerical Methods? *"Case #I"*

- Unfortunately, the integral in the previous slide cannot be evaluated in a closed form (i.e. analytically), and hence, numerical integration methods (i.e. here in ***ENGG 407***) are used instead.

- In such techniques, **tabulated values** of the integrals are obtained and recorded.

- So, you can remember the **z**, **t**, χ^2, and **f** tables (distributions) you used in ***ENGG 319***.

© Sameh Nassar

Why Numerical Methods? *"Case #II"*

Something You Know:

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

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Analytic Solution

Roots (zeros) of f(x)

But if you have (for example):

$$f(x) = e^{-x} - x = 0$$

Cannot be solved analytically

Approximate sol. technique is the only alternative!

© Sameh Nassar

Numerical Methods: Programming

- Although numerical methods are approximate, they employ "systematic" strategies to estimate the solution.
- In this case, the convergence "*quickness*" and the accuracy "*correctness*" of the approximate solution depends on the increment value, the used interval and the initial value (*Does this remind you with* ***ENGG 233****?*).

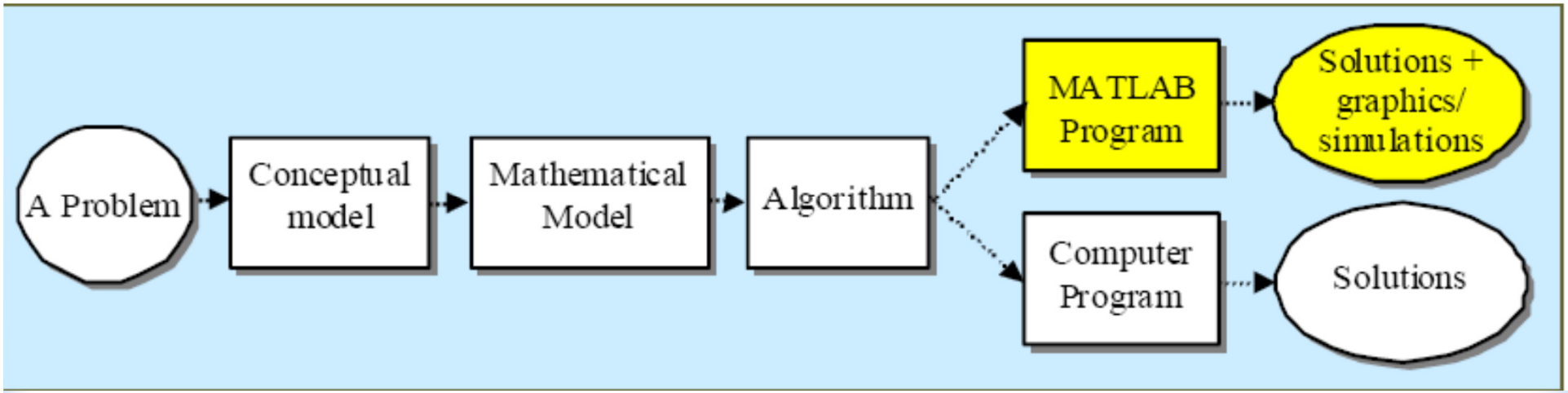
© Sameh Nassar

Numerical Methods: Programming

- So, in our case you need to implement an ***algorithm*** (methodology/steps of performing the numerical method) and a ***code*** (list of computer commands to execute the algorithm).
- Error analysis is necessary to evaluate the quality of the obtained approximate solution.
- Thus, what are the major differences between numerical and analytic Methods?

© Sameh Nassar

Numerical Methods for Solving Engineering Problems



Yingxu Wang (ENGG 407 2011 notes)

© Sameh Nassar

Numerical Methods: Errors

- Since Numerical methods involve an approximate solution of the problem, there will be always a discrepancy "an **error**" between the exact "analytical" solution and the numerical solution.

- In most situations, we don't know the exact solution, and hence, all we can do is to have an "estimate" or an "approximation" of the error.

© Sameh Nassar

Representation of Numbers

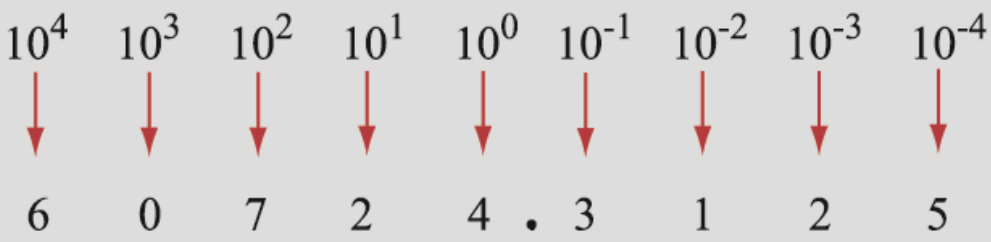
- Numbers can be represented in various forms (number systems).
- A number can be written by a sequence of digits that correspond to multiple powers of a certain ***base***.
- Any integer digit greater than "1" can be used as the ***base*** for a number system, e.g. 2, 8 (*octal*), 10, 16 (*hexadecimal*).
- If the ***base*** is larger than 10, the letters *A, B, C, D,…* are usually used to represent *10, 11, 12, 13,….*
- The familiar ***decimal system*** uses ***base 10***.
- However, computers use the ***binary system*** (***base 2***) due to its easy implementation in computer operations.

© Sameh Nassar

Representation of Numbers

Decimal Representation (*Base 10*):

- The decimal system uses ten "*decimal*" digits: 0, 1, 2, ..., 9.
- A number is written by a sequence of digits that correspond to multiples of powers of 10.



$$6\times10^4+0\times10^3+7\times10^2+2\times10^1+4\times10^0+3\times10^{-1}+1\times10^{-2}+2\times10^{-3}+5\times10^{-4}=60{,}724.3125$$

© Sameh Nassar

Representation of Numbers on Computers

Binary Representation (*Base 2*):

- The binary system uses two "*binary*" digits: ***0*** and ***1***.
- A number is written by a sequence of ***zeros*** and ***ones*** that correspond to multiples of powers of 2.

2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	2^{-1}	2^{-2}	2^{-3}	2^{-4}
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
1	1	1	0	1	1	0	1	0	0	1	1	0	1	0	0 .	0	1	0	1

$$1\times2^{15}+1\times2^{14}+1\times2^{13}+0\times2^{12}+1\times2^{11}+1\times2^{10}+0\times2^{9}+1\times2^{8}+0\times2^{7}+0\times2^{6}+1\times2^{5}$$
$$+1\times2^{4}+0\times2^{3}+1\times2^{2}+0\times2^{1}+0\times2^{0}+0\times2^{-1}+1\times2^{-2}+0\times2^{-3}+1\times2^{-4}=60{,}724.3125$$

Representation of Numbers on Computers

Base 10	Base 2			
	2^3	2^2	2^1	2^0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1

$$0*2^3 + 1*2^2 + 1*2^1 + 1*2^0$$
$$= 0 + 4 + 2 + 1$$
$$= \mathbf{6}$$

- Binary arithmetic is used by computers since transistors can be used as extremely fast switches where "***1***" and "***0***" refer to switch being "***on***" and "***off***", respectively.

© Sameh Nassar

Representation of Numbers on Computers

- For binary arithmetic in computers, each binary digit (i.e. ***1*** or ***0***) is called a ***bit***.
- Sometimes, numbers having a finite representation in one system (with a specific ***base***) may have infinite (or not exact) representation in another system with a different base. For Ex:

  0.1 (decimal) = 0.0001100110011….(binary).
- In these cases, round-off and truncation errors (to be discussed later) will occur.
- To accomoddate for larege and small numbers, real numbers are represnted using the ***floating point representation***.

© Sameh Nassar

Floating Point Representation

Decimal Floating Point Representation (*Scientific Notation*):

- It has the form: $d_0.d_1d_2...d_n \times 10^p$. (d's are decimal digits)
- The number $0.d_1d_2...d_n$ is called the ***mantissa***.
- The integer "**p**" (i.e. the power of 10) is called the ***exponent***.
- "**p**" represents the number's order of magnitude (provided the preceding number is smaller than 5), otherwise the number is said to be of the order of **p+1**.
- **For example:**

  2.9543×10^{-7} is of the order of 10^{-7} "written as $O(10^{-7})$"

  7.954326×10^{3} is of the order of 10^{4} "written as $O(10^{4})$"

© Sameh Nassar

Floating Point Representation

Decimal Floating Point Representation (*Scientific Notation*):

- For this form: $d_0.d_1d_2...d_n \times 10^p$, the number of significant digits is **n+1**.

standard notat.	*scientific notat.*	*order of magn.*	*# of sign. digits*
4519.23	4.51923×10^3	$O(10^3)$	6
732.5051	7.325051×10^2	$O(10^3)$	7
-0.00012	-1.2×10^{-4}	$O(10^{-4})$	2
-0.005612	-5.612×10^{-3}	$O(10^{-2})$	4

© Sameh Nassar

Floating Point Representation

Binary Floating Point Representation:

- It has the form: $\boldsymbol{1.b_1b_2...b_n} \times \mathbf{2}^{\mathbf{B_1B_2...B_m}}$. *(b's and B's are binary digits)*
- The binary digits $\boldsymbol{.b_1b_2...b_n}$ is the ***mantissa***.
- The binary digits "$\boldsymbol{B_1B_2...B_m}$" (i.e. the power of 2) is the ***exponent***.
- A decimal number is converted to binary floating point representation using a ***two-step approach***:

  (a) Normalize the decimal number with respect to the largest power of ***2*** that is smaller than the decimal number itself. Note that the new (*normalized*) decimal number will have the following form:

  $\boldsymbol{1.d_1d_2...d_j} \times \mathbf{2}^{\mathbf{D_1D_2...D_k}}$ *(where the d's and D's are decimal digits)*

- (b) Then, convert the mantissa and the power of 2 of the *normalized decimal number* from decimal to binary.

© Sameh Nassar

Floating Point Representation

Binary Floating Point Representation:

Textbook Examples:

Decimal Number	Normalized Decimal Number	Binary Floating Point Rep.
50	$50/2^5 \times 2^5 = 1.5625 \times 2^5$	1.1001×2^{101}
	$(50/32 \times 2^5)$	
0.3125	$0.3125/2^{-2} \times 2^{-2} = 1.25 \times 2^{-2}$	1.01×2^{-10}
	$(0.3125/0.25 \times 2^{-2})$	
1344	$1344/2^{10} \times 2^{10} = 1.3125 \times 2^{10}$	1.0101×2^{1010}
	$(1344/1024 \times 2^{10})$	

© Sameh Nassar

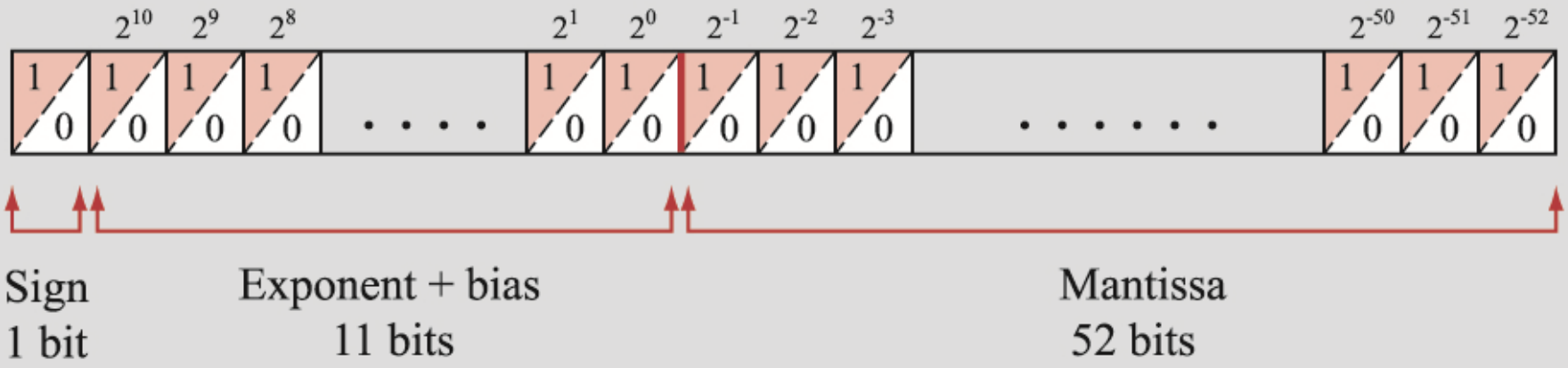
Storing Numbers in Computer Memory

- Once in binary floating point representation, numbers are stored in the computer.
- Computers store the values of the exponent and the mantissa separately, while the leading "**1**" in front of the decimal point (i.e. that leads the mantissa is not stored.
- The memory in the computer is organized in bytes (each byte is 8 bits).
- Computers store numbers and perform calculations in either ***single precision*** (using a string of 32 bits, i.e. 4 bytes) or ***double precision*** (using a string of 64 bits, i.e. 8 bytes).
- This is according to the IEEE (Institute of Electrical and Electronics Engineers) IEEE-754 standard.

© Sameh Nassar

Storing Numbers in Computer Memory

- In both cases (single & double precisions), the first bit stores the sign of the number (**0** for **+** and **1** for **-**).
- The following 8 bits in single precision (11 bits in double precision) are used to store the exponent.
- Then, the next 23 bits in single precision (52 bits in double precision) are used to store the mantissa.



© Sameh Nassar

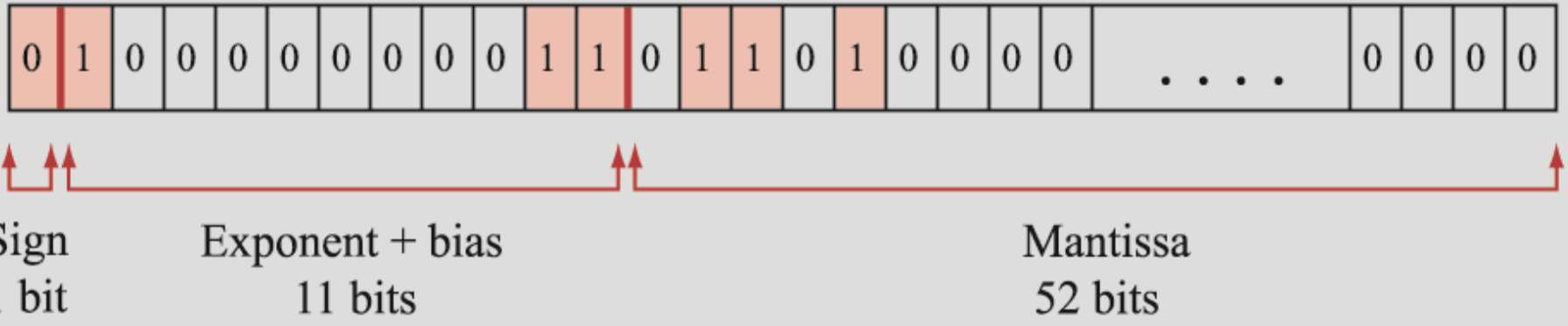
Storing Numbers in Computer Memory

- As can be seen from the previous slide, the exponent is stored with adding a bias (a constant which is positive in this case) to avoid using one of the bits for the sign of the exponent.
- In double precision, the used bias is 1023.
- Thus, the largest (i.e. positive) ***actual*** exponent that can be stored is 1024 (stored as 2047), **why?**.
- Also, the smallest (i.e. negative) ***actual*** exponent that can be stored is -1023 (stored as 0).

© Sameh Nassar

Storing Numbers in Computer Memory

- Here is how the decimal number 22.5 is stored (as a floating point representation) using double precision.
- Normalization: $22.5/2^4 \times 2^4 = 22.5/16 \times 2^4 = 1.40625 \times 2^4$
- Exponent with bias (11 bits): 4+1023 = 1027 ➡ 10000000011 (in binary).
- Mantissa (52 bits): .011010000….0000 (in binary).
- Sign (1 bit): 0 (positive).



© Sameh Nassar

Storing Numbers in Computer Memory

- Since a finite number of bits is used (either in single or double precision), not every number can be accurately represented in binary.

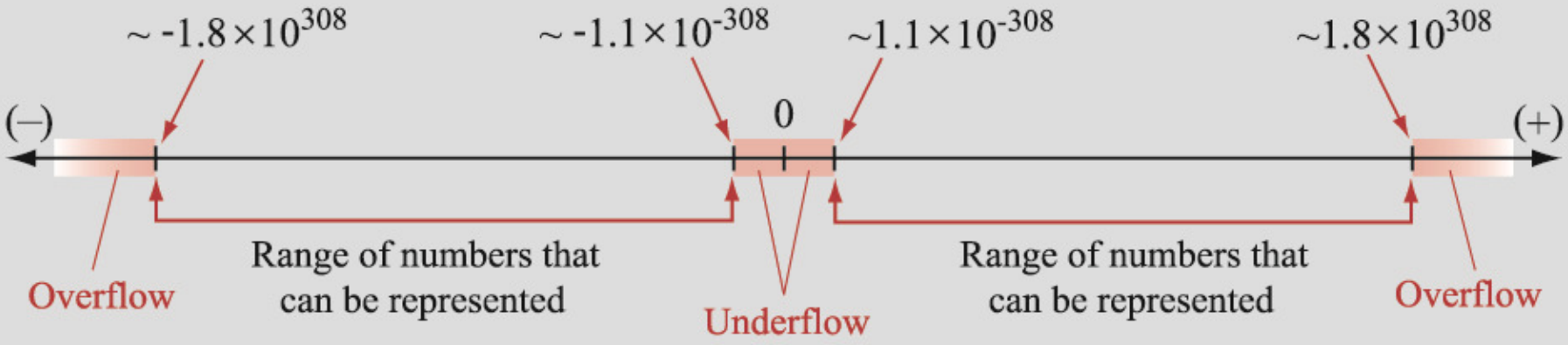
In double precision:

- The smallest positive number that can be expressed is 2^{-1023} (binary) $\approx 1.1 \times 10^{-308}$ (decimal), i.e. not zero!.
- The corresponding largest negative number is -1.1×10^{-308} (decimal).
- The largest positive number that can be expressed is 2^{1024} (binary) $\approx 1.8 \times 10^{308}$ (decimal), i.e. not ∞!.
- The corresponding smallest negative number is -1.8×10^{308} (decimal).

© Sameh Nassar

Storing Numbers in Computer Memory

- Numbers outside the ranges given in the previous slide will cause either ***overflow errors*** or ***underflow errors*** (depending on the case, see figure below).



- In addition to ***underflow*** and ***overflow*** errors, other errors will occur in case of infinite (or not exact) binary representation of some decimal numbers (*Recall the **0.1** example discussed earlier*).

© Sameh Nassar

Required Mathematical Refreshments

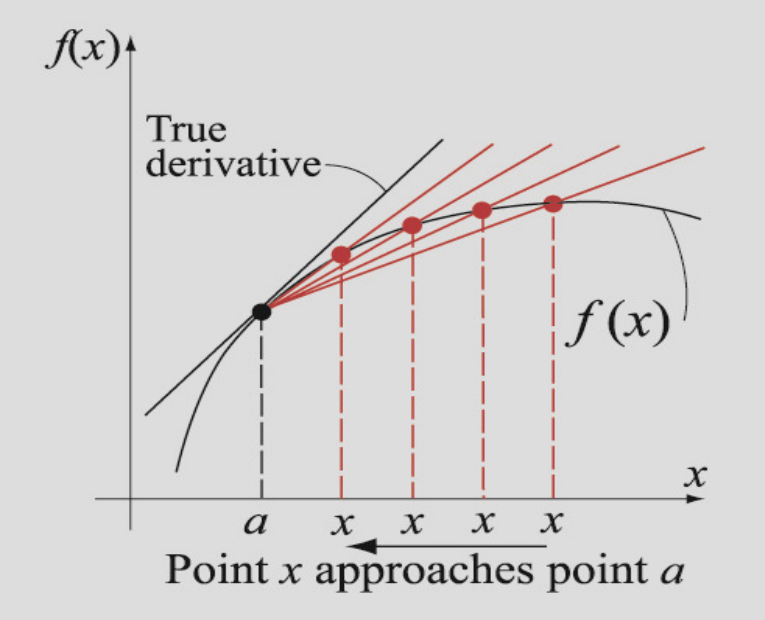
- Limit of a function.
- Continuity of a function.
- Intermediate value theorem.
- **Derivative of a function.**
- Chain rule for ordinary differentiation.
- **Mean value theorem for derivatives.**
- Integral of a function.
- Fundamental theorem of calculus.
- Mean value theorem for integrals.
- Derivatives of functions of two independent variables (Partial derivatives & Chain rule).

© Sameh Nassar

Required Mathematical Refreshments

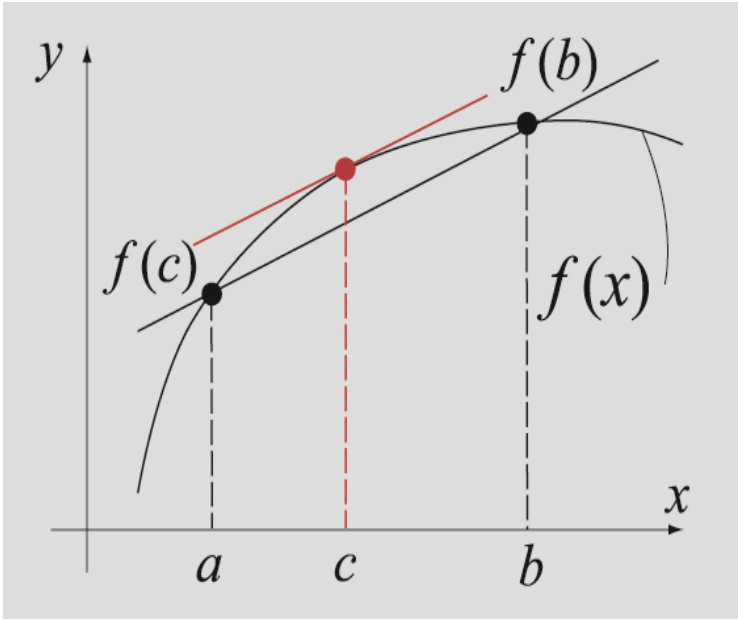
Derivative of a function:

$$f'(a) = \left.\frac{df(x)}{dx}\right|_{x=a} = \lim_{x \to a} \frac{f(x) - f(a)}{x - a}$$



Mean value theorem for derivatives:

$$f'(c) = \left.\frac{df(x)}{dx}\right|_{x=c} = \frac{f(b) - f(a)}{b - a}$$



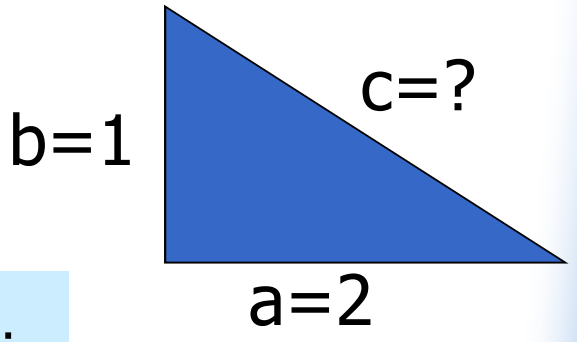
© Sameh Nassar

Errors Definition (1/4)

- **Truncation Errors:** Result when approximations are used to represent exact mathematical procedures. (*will be explained in Taylor Series discussion*)
- **Round-Off Errors:** Results when numbers having limited significant figures are used to represent exact numbers (This includes computers that can represent only quantities with a finite number of digits).

Ex:

$$c = \sqrt{a^2 + b^2} = \sqrt{2^2 + 1^2} = \sqrt{5}$$



c_1=2.2 , c_2= 2.24 , c_3=2.236 , c_4=2.2361 , ……

c_{14} =2.23606797749979

$(c_3)^2 = (2.236)^2 = 4.9997 \sim 5$ (but not exactly 5)

Errors Definition (2/4)

Total Error = Truncation Errors + Round-Off Errors

- **True Error (E_t):**

$$E_t = \text{True Value} - \text{Approximation}$$

- **True Relative Error (e_t):**

$$e_t = \frac{\text{True Error}}{\text{True Value}} = \frac{E_t}{\text{True Value}}$$

- **True Percent Relative Error (ε_t):**

$$\varepsilon_t = \frac{\text{True Error}}{\text{True Value}} 100\% = \frac{E_t}{\text{True Value}} 100\%$$

© Sameh Nassar

Errors Definition (3/4)

Total Error = Truncation Errors + Round-Off Errors

- **Approximate Error (E_a):**

  E_a = Current Approximation – Previous Approximation

- **Approximate Relative Error (e_a):**

$$e_a = \frac{\text{Approximate Error}}{\text{Approximation}} = \frac{E_a}{\text{Approximation}}$$

- **Approximate Percent Relative Error (ε_a):**

$$\varepsilon_a = \frac{\text{Approximate Error}}{\text{Approximation}} 100\% = \frac{E_a}{\text{Approximation}} 100\%$$

© Sameh Nassar

Errors Definition (4/4)

When we should stop our "approximation"? i.e. what will be our latest computed ε_a?

This is obtained by using a prespecified percent tolerance (acceptable relative error) ε_s , such that:

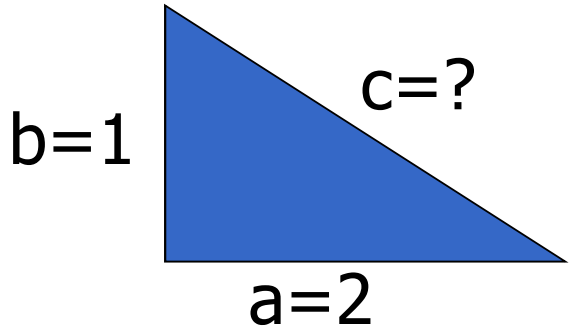
$$|\varepsilon_a|_{last} < \varepsilon_s$$

We usually use $|\varepsilon_a|$

© Sameh Nassar

Example #1

Compute the value of ***c*** with approximating your answer to:
(a) three decimals.
(b) four decimals.



<u>Then, obtain the following:</u>

(1) the true percent relative error for (a) and (b)
(2) the approximate percent relative error using (a) and (b)

Assume the true value of ***c*** is the one with 14 decimals.

© Sameh Nassar

Example #1 (Sol.)

$$c = \sqrt{a^2 + b^2} = \sqrt{2^2 + 1^2} = \sqrt{5}$$

c_{14} = 2.23606797749979 (assumed true value)

(a) c_3 = 2.236 (b) c_4 = 2.2361

$$(1)\ |\varepsilon_{t_3}| = \left|\frac{2.236 - 2.23606797749979}{2.23606797749979}\right| 100\% = 0.003\%$$

$$(1)\ |\varepsilon_{t_4}| = \left|\frac{2.2361 - 2.23606797749979}{2.23606797749979}\right| 100\% = 0.001\%$$

$$(2)\ |\varepsilon_{a_{4,3}}| = \left|\frac{2.2361 - 2.236}{2.2361}\right| 100\% = 0.005\%$$

→ **Comment!**

© Sameh Nassar

Example #2

In **Example #1**, use an acceptable tolerance of 0.002% to "terminate" computing the value of ***c*** (start from one decimal).

$c_1=2.2$, $c_2=2.24$ ⇒ $|\varepsilon_{a_{2,1}}|=\left|\frac{2.24-2.2}{2.24}\right|100\%=1.667\%$

$c_2=2.24$, $c_3=2.236$ ⇒ $|\varepsilon_{a_{3,2}}|=\left|\frac{2.236-2.24}{2.236}\right|100\%=0.179\%$

$c_3=2.236$, $c_4=2.2361$ ⇒ $|\varepsilon_{a_{4,3}}|=\left|\frac{2.2361-2.236}{2.2361}\right|100\%=0.005\%$

$c_4=2.2361$, $c_5=2.23607$ ⇒ $|\varepsilon_{a_{5,4}}|=\left|\frac{2.23607-2.2361}{2.23607}\right|100\%=0.001\%$

© Sameh Nassar

Rules for Significant Figures

- Engineers usually work with numbers that obtain several digits and hence should determine how many of them are significant.
- A ***significant digit*** in a number is the least *significant figure* that is considered reliable in the number as a result of measurements or calculations.
- The number of *significant figures* in a result indicates the number of digits that can be used with confidence.
- A common mistake is to show too many figures in the answer, which gives the impression that the answer is more accurate than it really is.

© Sameh Nassar

Rules for Significant Figures

Rule #1:

- Same significant digit must be used in a coherent set of engineering operations and data processing.
- When different significant digits are involved in an operation, a truncation and rounding operation must be carried out in processing the result.

Rule #2:

- The rule of rounding in significant digits processing is to truncate 4 and round up 5. *For Example*:

6.2548 ⇒ 6.3 (for 1 significant decimal digit)

6.2548 ⇒ 6.25 (for 2 significant decimal digits)

6.2548 ⇒ 6.255 (for 3 significant decimal digits)

© Sameh Nassar

Rules for Significant Figures

Rule #3:

- The result of *addition* and *subtraction* should show significant digits only as far to the right as is seen in the least precise number in the calculation.
- The insignificant digits following the least significant figure should be rounded to the least significant figure.

Examples:

1725.463 - 189.2 = 1536.263 ⇒ 1536.3

23578.3 + 0.1892 = 23578.4892 ⇒ 23578.5

3.510 + 2.246 + 0.0192 = 5.7752 ⇒ 5.775

© Sameh Nassar

Rules for Significant Figures

Rule #4:

- The result of *multiplication or division* should show significant digits only as far to the right as that are contained in the number with the fewest significant digits.
- The insignificant digits following the least significant figure should be rounded to the least significant figure.

Examples:

$2.43 \times 17.675 = 42.950250 \Rightarrow 42.95$

$75.22 \div 25.1 = 2.9968127 \Rightarrow 3.0$

$75.220 \div 25.100 = 2.9968127 \Rightarrow 2.997$

© Sameh Nassar

Taylor Series

- "is a mathematical formulation that is used widely in numerical methods to express functions in an approximate fashion".

- "provides a means to predict a function value at one point in terms of the function value and its derivatives at another point".

© Sameh Nassar

Taylor Series

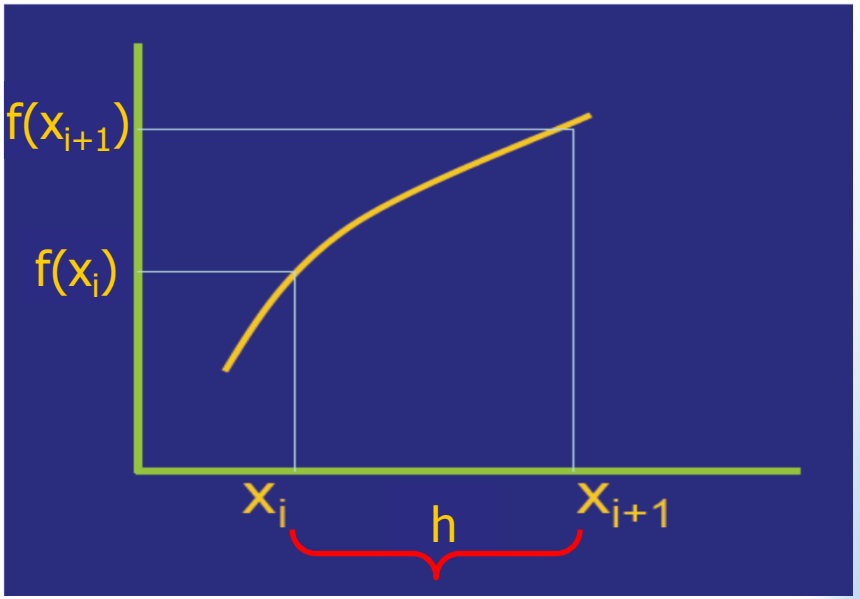
- For a function ***f(x)*** that depends on only one independent variable ***x***, the value at point x_{i+1} can be approximated by the following Taylor Series:

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

The term R^n is called the "**Remainder**"

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

where ξ is a value of x that lies between x_i and x_{i+1}



© Sameh Nassar

Taylor Series

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

$$f(x_{i+1}) = \sum_{k=0}^{\infty} \frac{f^{(k)}(x_i)}{k!}h^k$$

$f(x_{i+1}) \cong f(x_i)$ zero order approximation

$f(x_{i+1}) \cong f(x_i) + f'(x_i)h$ 1st order approximation

$f(x_{i+1}) \cong f(x_i) + f'(x_i)h + \frac{f''(x_i)}{2!}h^2$ 2nd order approximation

$\vdots$

$f(x_{i+1}) \cong f(x_i) + f'(x_i)h + \frac{f''(x_i)}{2!}h^2 + \ldots + \frac{f^{(n)}(x_i)}{n!}h^n$ nth order approx.

Example #3

Use Taylor series to expand the function:

f(x) = sin(x) with an expansion point of *x=0.2* to evaluate *f(x)* at *x=0.8*

In your expansion, use zero order, 1st order, 2nd order, 3rd order, and 4th order approximations.

Then, estimate the true relative error for each case.

Hint: all x values are in radians

© Sameh Nassar

Example #3 (Sol.”1/5”)

$x_i = 0.2, \quad x_{i+1} = 0.8$ ⇒ $h = x_{i+1} - x_i = 0.6$

Sin(0.8) = 0.717356090899523 (assumed true value)

f(x)=sin(x) , f′(x)=cos(x) , f″(x)=-sin(x) , $f^{(3)}(x)$=-cos(x) , $f^{(4)}(x)$=sin(x)

$$f(x_{i+1}) \cong f(x_i) + f'(x_i)h + \frac{f''(x_i)}{2!}h^2 + \frac{f^{(3)}(x_i)}{3!}h^3 + \frac{f^4(x_i)}{4!}h^4$$

zero order approx.

1st order approx.

2nd order approx.

3rd order approx.

4th order approx.

© Sameh Nassar

Example #3 (Sol."2/5")

$$\sin(0.8) \cong \sin(0.2) + \cos(0.2) * 0.6 - \frac{\sin(0.2)}{2!} * 0.6^2 - \frac{\cos(0.2)}{3!} * 0.6^3 + \frac{\sin(0.2)}{4!} * 0.6^4$$

Zero order approx.

$$\sin(0.8) \cong \sin(0.2) = 0.198669330795061$$

$$|\varepsilon_{t_0}| = \left| \frac{0.717356090899523 - 0.198669330795061}{0.717356090899523} \right| 100\% = 72.3\%$$

1st order approx.

$$\sin(0.8) \cong \sin(0.2) + \cos(0.2) * 0.6 = 0.786709277499806$$

$$|\varepsilon_{t_1}| = \left| \frac{0.717356090899523 - 0.786709277499806}{0.717356090899523} \right| 100\% = 9.7\%$$

© Sameh Nassar

Example #3 (Sol."3/5")

2nd order approx.

$$\sin(0.8) \cong \sin(0.2) + \cos(0.2) * 0.6 - \frac{\sin(0.2)}{2!} * 0.6^2 = 0.750948797956695$$

$$|\varepsilon_{t_2}| = \left| \frac{0.717356090899523 - 0.750948797956695}{0.717356090899523} \right| 100\% = 4.7\%$$

3rd order approx.

$$\sin(0.8) \cong \sin(0.2) + \cos(0.2) * 0.6 - \frac{\sin(0.2)}{2!} * 0.6^2 - \frac{\cos(0.2)}{3!} * 0.6^3$$

$$= 0.715666640115441$$

$$|\varepsilon_{t_3}| = \left| \frac{0.717356090899523 - 0.715666640115441}{0.717356090899523} \right| 100\% = 0.24\%$$

© Sameh Nassar

Example #3 (Sol."4/5")

4th order approx.

$$\sin(0.8) \cong \sin(0.2) + \cos(0.2) * 0.6 + \frac{\sin(0.2)}{2!} * 0.6^2 - \frac{\cos(0.2)}{3!} * 0.6^3 - \frac{\sin(0.2)}{4!} * 0.6^4$$

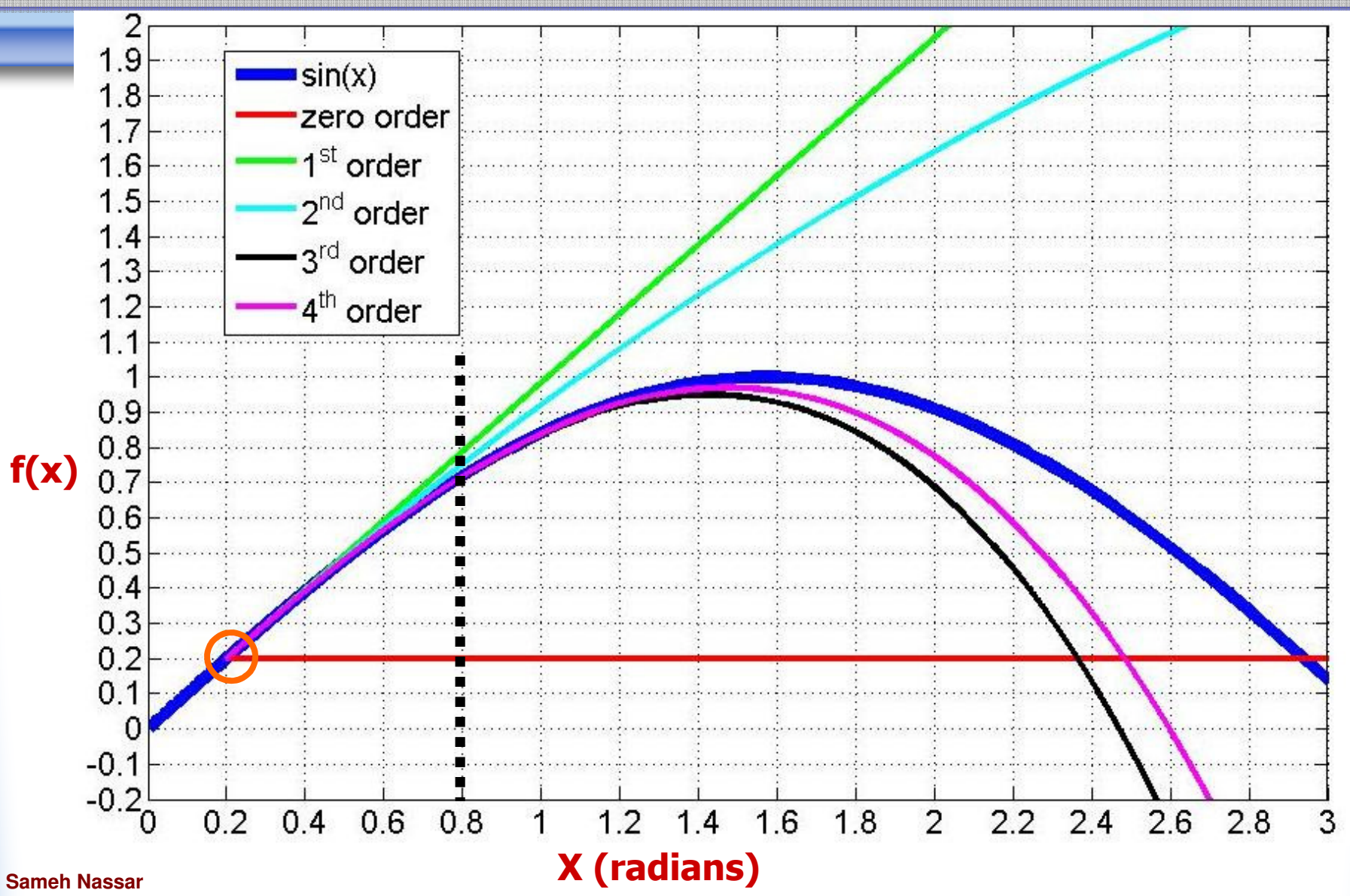
$$= 0.716739215540704$$

$$|\varepsilon_{t_4}| = \left| \frac{0.717356090899523 - 0.716739215540704}{0.717356090899523} \right| 100\% = 0.086\%$$

Comment!

© Sameh Nassar

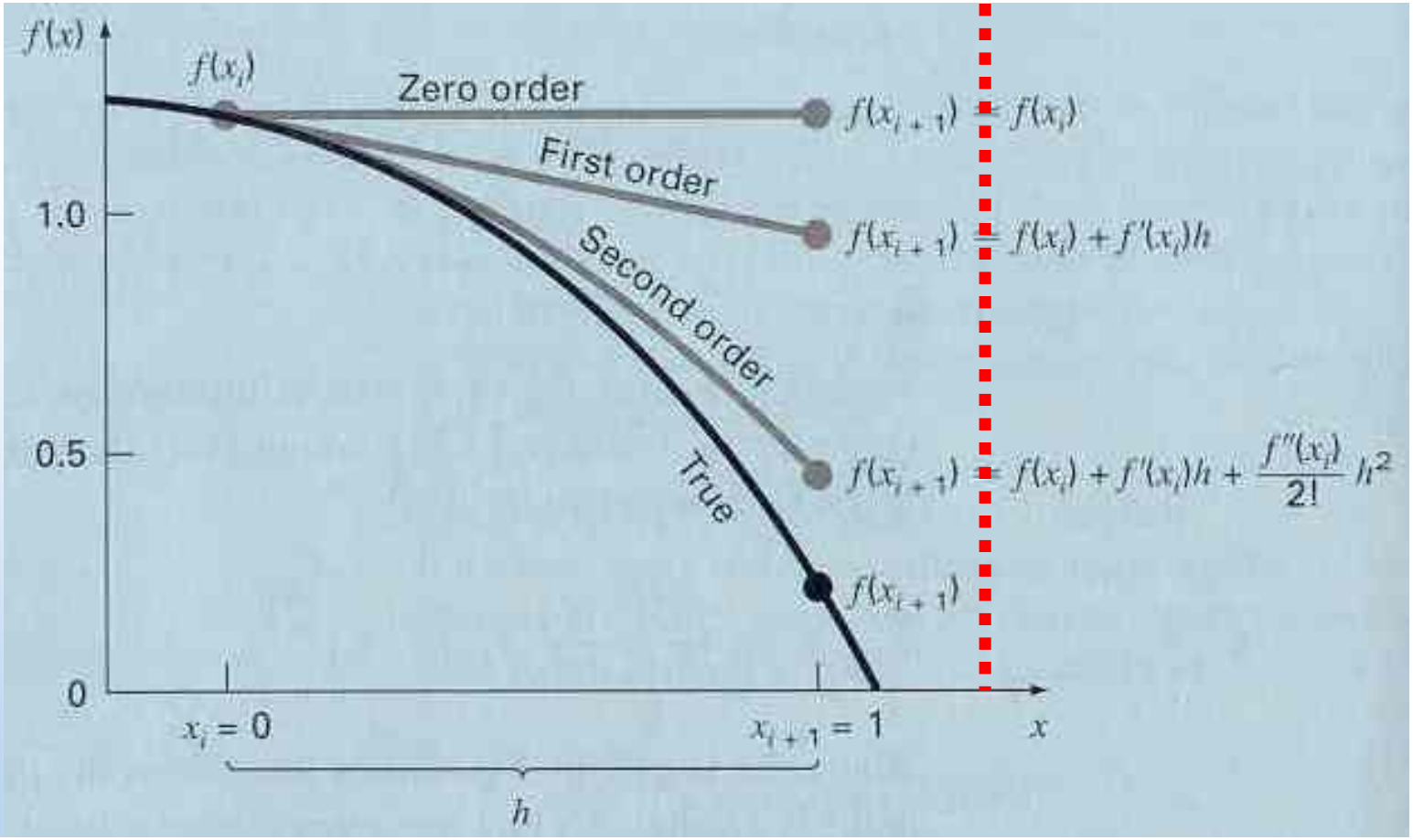
Example #3 (Sol."5/5")



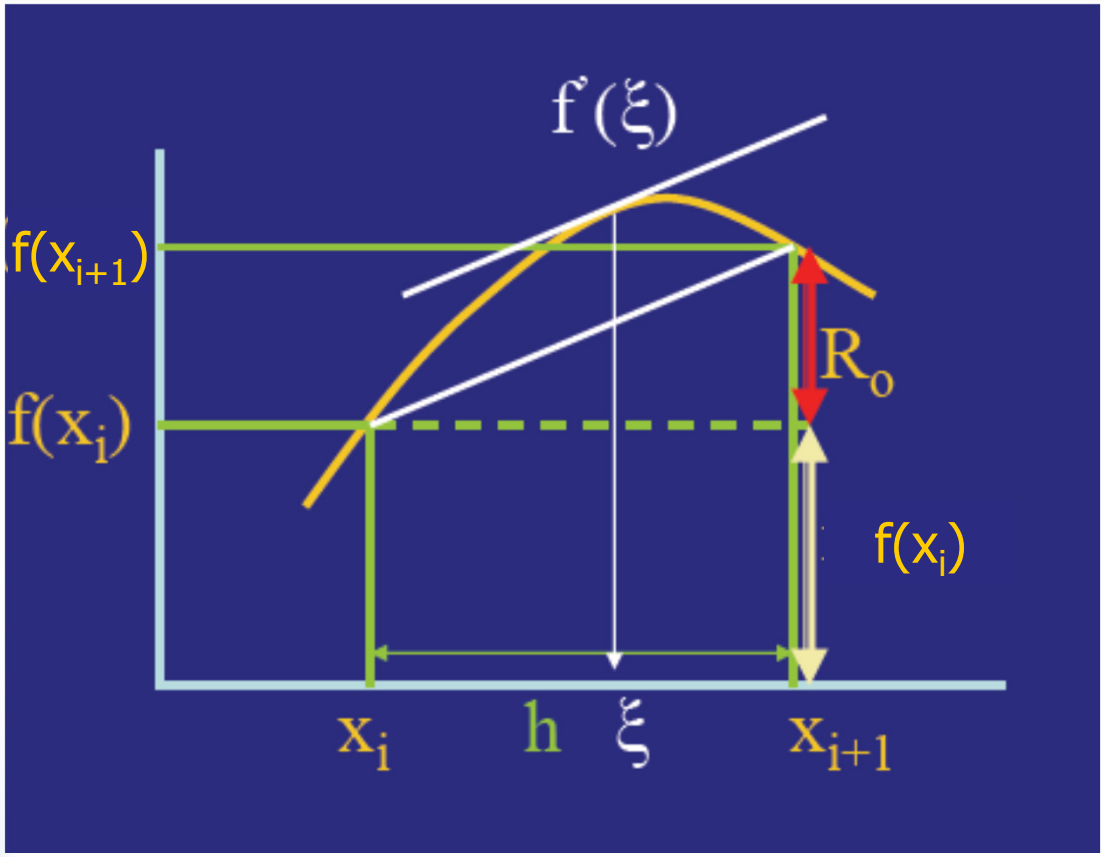
© Sameh Nassar

Finite Number of Derivatives

$$f(x) = -0.1x^4 - 0.15x^3 - 0.5x^2 - 0.25x + 1.2$$



Truncation Error (1/2)



$$\frac{R_0}{h} = f'(\xi)$$

$$R_0 = f'(\xi)h = O(h)$$

Similarly:

$$R_1 = \frac{f''(\xi)}{2!}h^2 = O(h^2)$$

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

Truncation Error

© Sameh Nassar

Truncation Error (2/2)

- **In general, we can usually assume that the truncation error is decreased by:**

  (1) adding more terms to the Taylor series

  (2) Reducing the step size h

- The Taylor series becomes more accurate when the approximated function behaves as a "straight-line" over the interval of interest (which is usually not the case!).

- To overcome this problem, we should perform (1) and (2) above.

© Sameh Nassar

Why Taylor Series?

Suppose that you have to deal with:

$$f(x) = ln\left[x^4 + cos(x) + ... - \sum_{k=0}^{\infty} \frac{5}{x^k} sin(x) + 2x \right]$$

- Difficult to manipulate for further processing
- Difficult to visualize (without plotting) in terms of slope and curvature for example.

Remember:

In Taylor series expansion, we evaluate the *f(x)* value within a specific interval (not for the whole range of *x* values)

© Sameh Nassar

Taylor Series: Remarks

- Sometimes, Taylor Series expansion can be written using the following format:

$$f(x) = f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \ldots + \frac{f^{(n)}(a)}{(n)!}(x-a)^n + R_n$$

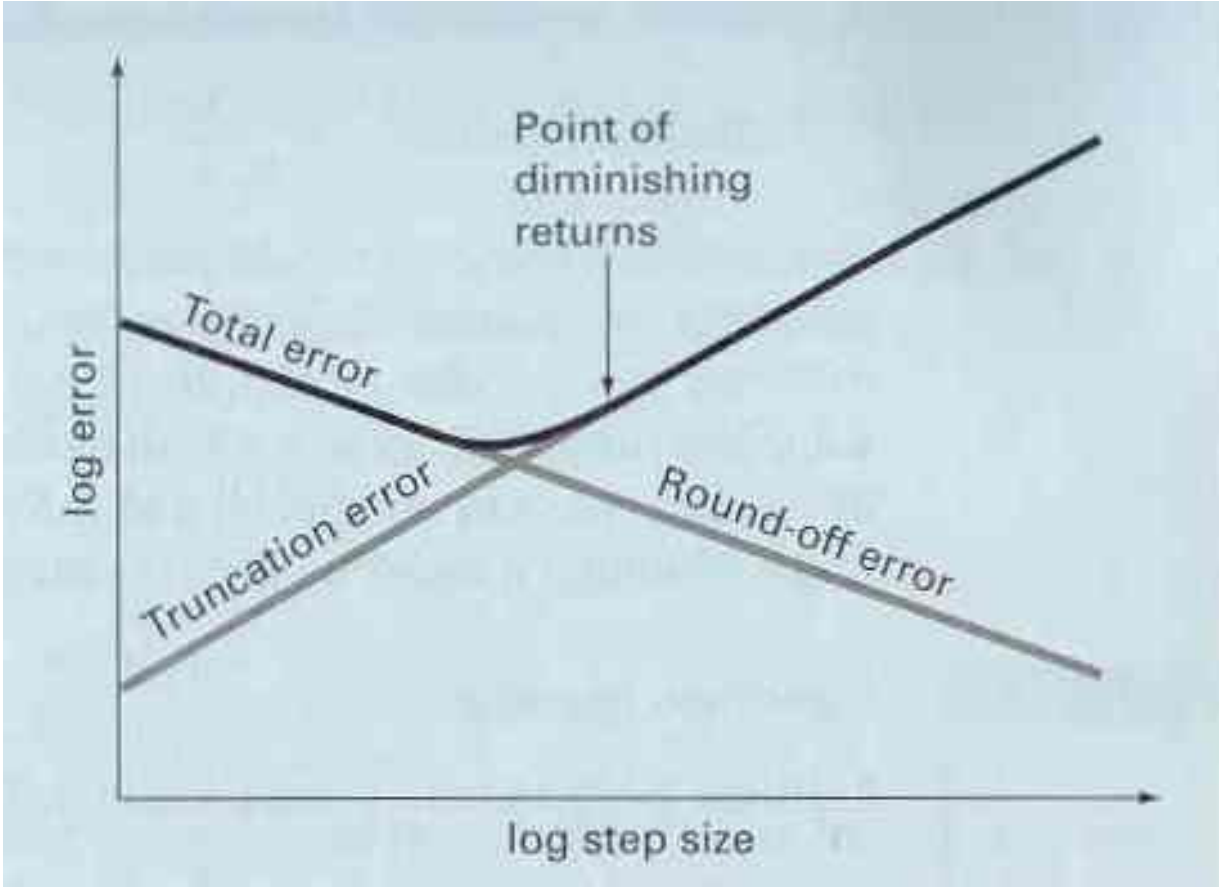
i.e. replacing the symbols x_{i+1} by x and x_i (point of expansion) by a.

- For a function that is a function of two or more (independent) variables, Taylor Series can be applied in the same manner as for one variable, however, the differentiation will involve partial derivatives.

© Sameh Nassar

Total Numerical Error

Total Numerical Error =
***Sum* (Truncation Errors + Round-Off Errors)**



© Sameh Nassar

Textbook Readings

- **Sections Covered:**

  1.1, 1.2, 1.3, 1.4, 2.1, 2.2, 2.7

- **Required Background Sections:**

  2.4.1, 2.4.3, 2.4.4, 2.4.5, 2.6

© Sameh Nassar