ESO 208A: Computational Methods in Engineering

Introduction, Error Analysis

Saumyen Guha

Department of Civil Engineering
IIT Kanpur

What is *Computational Methods* or *Numerical Methods* in Engineering?

Formulation of *mathematical problems* in such a way that *numerical answers* can be computed using *arithmetic operations* in a computer

Computer can only perform: + - × /

All kinds of problems can be solved: linear and non-linear systems of equations; approximation and interpolation of data; differentiation and integration, ODE, PDE

What is *Numerical Analysis* ?

Mathematical method to analyze whether the solution obtained from a *Numerical Method* represents the solution of the original *mathematical problem* !

A way to test for *convergence!*

Convergence can be *conditional* or *unconditional*!

A method cannot be used for a given problem without the analysis!

Example 1: Calculate the square root of 2.

- ✓ Mathematical Problem: $x^2 = 2$, $x = ?$ or Calculate the root of $x^2 – 2 = 0$
- ✓ Square root of 2 (an irrational number) cannot be computed using simple arithmetic operations!
- ✓ Generate a *Cauchy sequence* that converges to the square root of 2.
- ✓ Some Numerical Methods:
  - ✓ $x_{n+1} = \frac{2}{x_n}$
  - ✓ $x_{n+1} = \frac{1}{2}\left(x_n + \frac{2}{x_n}\right)$
  - ✓ $x_{n+1} = \frac{x_n+2}{x_n+1}$

Let's start all the methods with $x_0 = 1$ and compute for five iterations:

- ✓ $x_{n+1} = \frac{2}{x_n}: x_1 = 2, x_2 = 1, x_3 = 2, x_4 = 1, x_5 = 2$

- ✓ $x_{n+1} = \frac{1}{2}\left(x_n + \frac{2}{x_n}\right):\ x_1 = 1.5, x_2 = 1.416667, x_3 = 1.414216,$

  $x_4 = 1.414214,\ x_5 = 1.414214$

- ✓ $x_{n+1} = \frac{x_n+2}{x_n+1}: x_1 = 1.5, x_2 = 1.4, x_3 = 1.416667,\ x_4 =$

  $1.413793,\ x_5 = 1.414286$

Could we have predicted this without having to perform computations?

Test for convergence of sequence (MTH 101):

- ✓ Sequence: $x_{n+1} = \frac{1}{2}\left(x_n + \frac{2}{x_n}\right)$
- ✓ $x_n^2 + 2 = 2x_n x_{n+1}$
- ✓ Since the root is real, $\left(x_n - \sqrt{2}\right)^2 \geq 0$
- ✓ Therefore, $x_n^2 + 2 = 2x_n x_{n+1} \geq 2\sqrt{2}x_n$
- ✓ $x_{n+1} \geq \sqrt{2}$, the sequence is bounded!
- ✓ for $n \geq 2,\ x_n - x_{n+1} = x_n - \frac{1}{2}\left(x_n + \frac{2}{x_n}\right) = \frac{1}{2}\left(\frac{x_n^2 - 2}{x_n}\right) \geq 0$, monotonically decreasing sequence!
- ✓ Monotonically decreasing sequence and bounded:- a *Cauchy sequence* that converges to the lower bound $\sqrt{2}$

Example 2: Solve the problem of the radioactive decay:

$$\frac{dN}{dt} = -\lambda N \quad \text{at } t = 0, N = N_0$$

True Solution:

$$N = N_0 e^{-\lambda t}$$

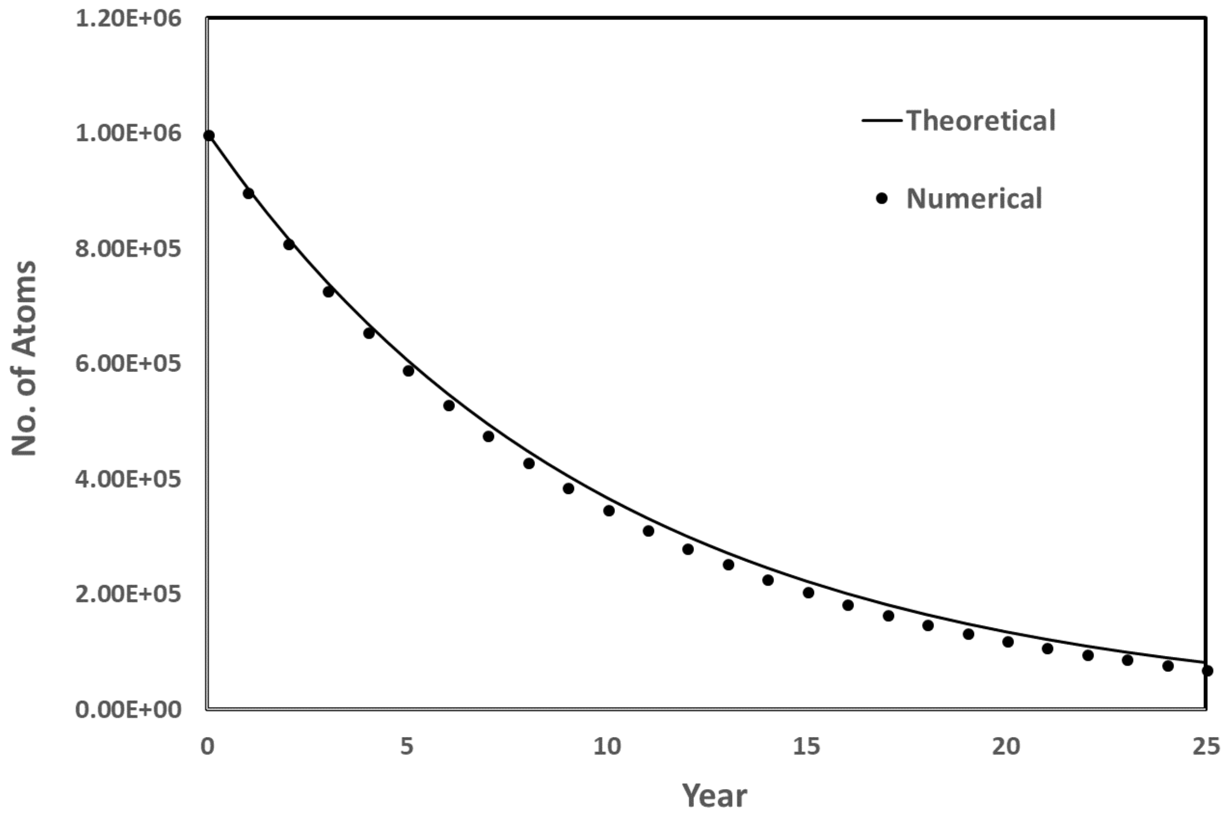
A Numerical Method: Euler Method (MTH 102)

$$N_{i+1} = N_i - \lambda h N_i \quad \text{where, } h = \text{time step } \Delta t$$

Consider two different elements with $\lambda = 0.1$ and 2.2

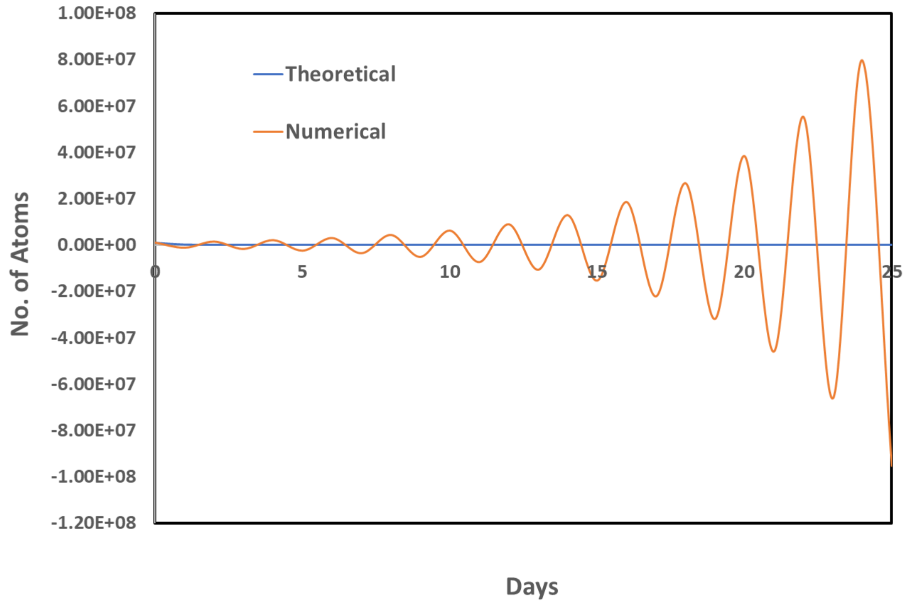
Let us compare true analytical solutions with the numerical solutions for $h = \Delta t = 1$ for these two elements.

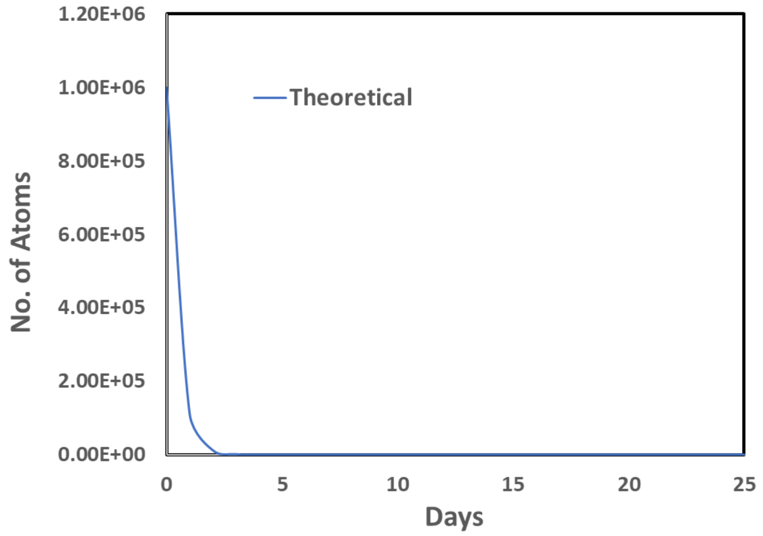
$\lambda = 0.1$ /day, $N_0 = 10^6$, $h = \Delta t = 1$, solution for 25 days



There is some error but the numerical solution *behaves* like the true solution

λ = 2.2 /day, $N_0 = 10^6$, $h = \Delta t = 1$, solution for 25 days

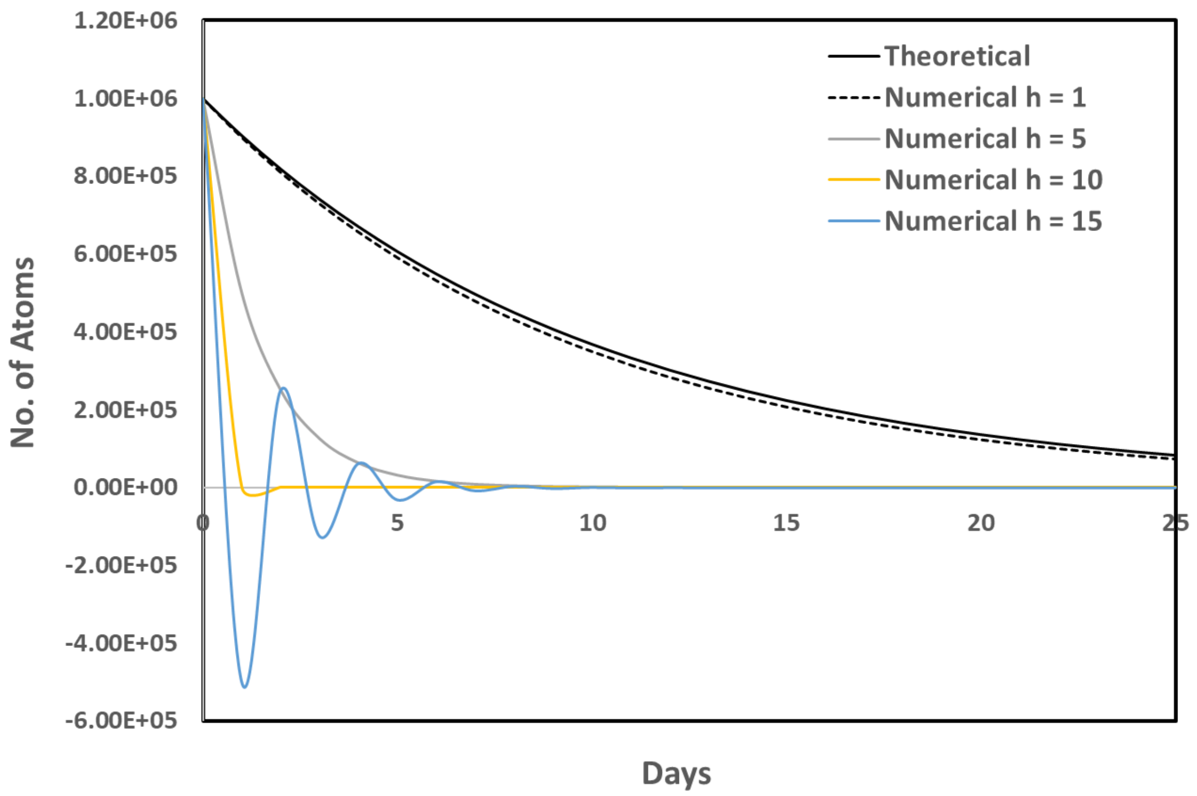




The error can grow to make the solution unacceptable!

Numerical analysis could have told me this saving much time of programming and computation!

$\lambda = 0.1$ /day, $N_0 = 10^6$, *varying h*, solution for 25 days



Numerical analysis could have told me this saving much time of programming and computation!

Take away message:

Learning *Numerical Analysis* is as important as the *Numerical Method* for you to be able to apply the method for a given problem!!

This is true irrespective of whether you are writing the program or using an existing program or package or software!

Errors and Error Analysis

Accuracy, Precision what are they?

✓ Accuracy:

how closely a computed or measured value agrees with the True value

opposite sense: inaccurate

✓ Precision:

How closely individual computed or measured values agree with each other

Opposite sense: imprecise

Define Error:

True Value (a) = Approximate Value ($\tilde{a}$) + Error (ε)

Absolute Error: $\varepsilon = (a - \tilde{a})$

Relative Error: $e = \frac{\varepsilon}{a} = \frac{(a-\tilde{a})}{a}$

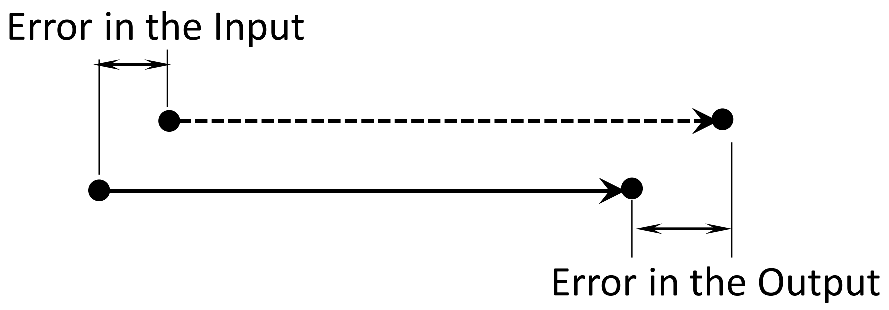
- ✓ Relative error is often expressed as (%) by multiplying (e) with 100.
- ✓ Absolute error can have sign as well as | . |
- ✓ If the error is computed with respect to the true value (if known), a prefix 'True' is added.
- ✓ For an iterative process, the true value 'a' is replaced with the previous iteration value and a prefix 'approximate' is added. This is used for testing convergence of the iterative process.

Sources of Error in computation?

- ✓ Errors in the Input data: initial and boundary conditions, measured values of the parameters and constants in the model
- ✓ Round-off error: irrational numbers, product and division of two numbers, limited by the machine capability
- ✓ Truncation error: truncation of an infinite series, often arises in the design of the numerical method through approximation of the mathematical problem.

Error Analyses:

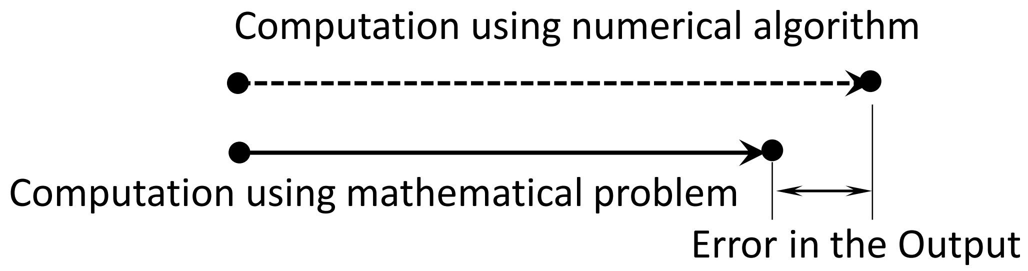
✓ **Forward Error Analysis:**



Both computations are using the original Mathematical Model

How an error in the given input propagates through the system model. (Robustness of the model)

✓ **Backward Error Analysis:**


Computation using numerical algorithm
Computation using mathematical problem
Error in the Output

Quantification of error resulting from interaction between round-off error and truncation error in the algorithm. (Robustness of the algorithm)

Forward Error Analysis:

Single Variable Function: $y = f(x)$. If an error is introduced in x, what is the error in y?

$$\Delta x = x - \tilde{x} \qquad \Delta y = y - \tilde{y} = f(x) - f(\tilde{x})$$

$$f(x) = f(\tilde{x} + \Delta x) = f(\tilde{x}) + \Delta x f'(\tilde{x}) + \frac{\Delta x^2}{2!} f''(\tilde{x}) + ..$$

Assuming the error to be small, the 2nd and higher order terms are neglected. (a first order approximation!)

$$\Delta y = f(x) - f(\tilde{x}) \approx \Delta x f'(\tilde{x})$$

Condition Number of the Problem (C_p):

$$C_p = \frac{Relative\ Error\ in\ y}{Relative\ Error\ in\ x} = \left|\frac{\Delta y / y}{\Delta x / x}\right| \approx \left|\frac{\Delta x f'(\tilde{x}) / f(x)}{\Delta x / x}\right| = \left|\frac{x f'(\tilde{x})}{f(x)}\right|$$

Also: $C_p = \left|\frac{\Delta y / y}{\Delta x / x}\right| = \left|\frac{\left(f(x) - f(\tilde{x})\right) / f(x)}{\Delta x / x}\right| = \left|\frac{x}{f(x)} \frac{f(\tilde{x} + \Delta x) - f(\tilde{x})}{\Delta x}\right|$

As $\Delta x \to 0$,

$$C_p = \left|\frac{x f'(\tilde{x})}{f(x)}\right|$$

$C_p < 1$: problem is well-conditioned, error is attenuated

$C_p > 1$: problem is ill-conditioned, error is amplified

$C_p = 1$: neutral, error is translated

Examples of Forward Error Analysis and C_p:

- ✓ Problem 1: $y = e^x$; $\Delta y = \Delta x e^x$; $C_p = x$. The problem is well-conditioned for $0 \leq |x| < 1$; neutral at $|x| = 1$ and ill-conditioned for $|x| > 1$.
- ✓ Problem 2: Solve the following system of equations:

$$x + \alpha y = 1; \qquad \alpha x + y = 0$$

Solving: $x = \frac{1}{1-\alpha^2} = x(\alpha)$; $x'(\alpha) = \frac{2\alpha}{(1-\alpha^2)^2}$

$$C_p = \left| \frac{\alpha \frac{2\alpha}{(1-\alpha^2)^2}}{\frac{1}{1-\alpha^2}} \right| = \left| \frac{2\alpha^2}{1-\alpha^2} \right|$$

well-conditioned for $\alpha << 1$ and ill-conditioned for $\alpha \approx 1$.

Forward Error Analysis:

Function of Multiple Variables: $y = f(X)$, where X is a vector, $X = \{x_0, x_1, \ldots\ldots x_n\}$. If error is introduced in the x_i's, what is the error in y?

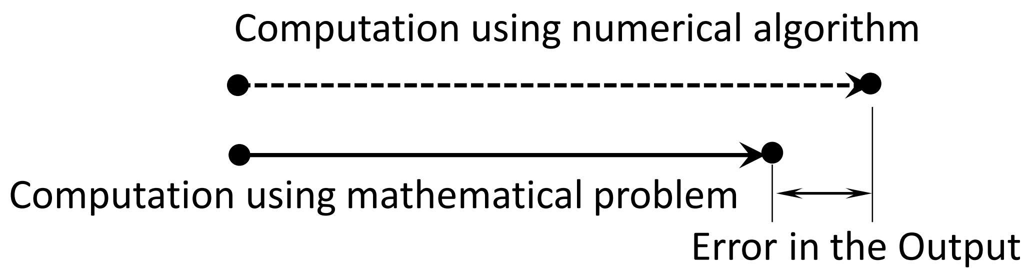
$$\Delta X = X - \tilde{X} = \begin{bmatrix} x_0 - \tilde{x}_0 \\ x_1 - \tilde{x}_1 \\ \vdots \\ x_n - \tilde{x}_n \end{bmatrix} = \begin{bmatrix} \Delta x_0 \\ \Delta x_1 \\ \vdots \\ \Delta x_n \end{bmatrix}$$

$$f(X) = f(\tilde{X}) + \sum_{i=1}^{n} \Delta x_i \left.\frac{\partial f}{\partial x_i}\right|_{\tilde{X}} + HOT$$

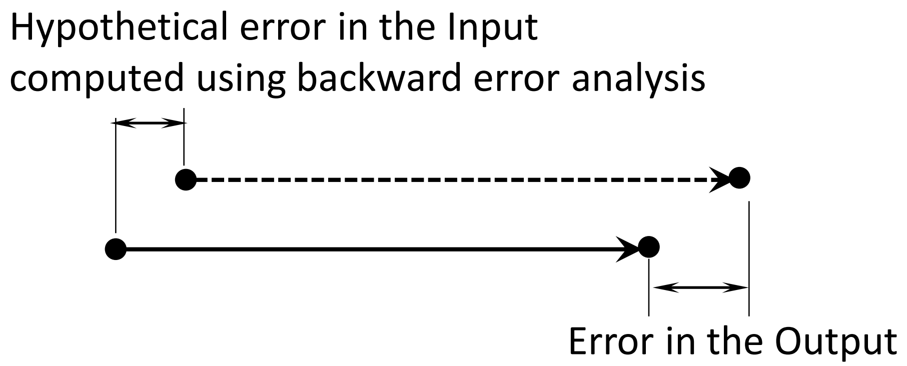
$$\Delta y = y - \tilde{y} = f(X) - f(\tilde{X}) \approx \sum_{i=1}^{n} \Delta x_i \left.\frac{\partial f}{\partial x_i}\right|_{\tilde{X}}$$

$$\text{An upper bound of } \Delta y = \sum_{i=1}^{n} |\Delta x_i| \left| \left.\frac{\partial f}{\partial x_i}\right|_{\tilde{X}} \right|$$

Bacward Error Analysis:


Computation using numerical algorithm
Computation using mathematical problem
Error in the Output

What is the perturbation required in the input in order to explain the error in the output if the computation is carried out by true mathematical function without any error.


Hypothetical error in the Input
computed using backward error analysis
Error in the Output

Both computations are using the original Mathematical Model

- ✓ Floating point representation of a number = $m \times b^q$
  - ✓ m = mantissa; $1/b \leq m \leq 1$
  - ✓ b = base; 2 for binary, 10 for decimal, 16 for hexadecimal
  - ✓ $q \in \mathbb{Q}$ (set of rationals)
- ✓ If a machine (computer) rounds a number off to 't' decimal places:

$$a = m \times 10^q; \qquad \tilde{a} = \tilde{m} \times 10^q$$

$$|m - \tilde{m}| \leq 0.5 \times 10^{-t}$$

- ✓ Machine round-off unit(u) is the upper bound of the relative error in one round-off operation

$$\left|\frac{a - \tilde{a}}{a}\right| \leq \frac{0.5 \times 10^{-t} \times 10^q}{m \times 10^q} \leq 0.5 \times 10^{1-t} = u$$

- ✓ Condition Number of Algorithm (C_a): change necessary in the input data in order to explain the error in the final result expressed in terms of u.

- ✓ Consider one floating point operation $fl(x \text{ op } y)$.
- ✓ 'op' can be any of +, -, ×, /.
- ✓ By definition of u:

$$\left|\frac{fl(x \text{ op } y) - (x \text{ op } y)}{(x \text{ op } y)}\right| \leq u$$

Let's consider the op as × :

$$fl(x \times y) = x \times y(1 + \delta) \leq x \times y(1 + u) \text{ where } |\,\delta\,| \leq u$$

- ✓ Example: A machine with $t = 2$, $u = 0.5 \times 10^{1-2} = 0.05$

$x = 0.30$, $y = 0.51$, $x \times y = 0.153$; $fl(x \times y) = 0.15$

Relative error = $|(0.15-0.153)/0.153| = 0.02 \leq u$ (= 0.05)

For this operation: $\delta = -0.01$; $|\,\delta\,| = 0.01 \leq u$ (= 0.05)

- ✓ Multiple Floating Point Operations:

$$fl(x_1 \times x_2 \times x_3 \times \cdots \times x_n)$$
$$= x_1 \times x_2(1+\delta_1) \times x_3(1+\delta_2) \times \cdots x_n(1+\delta_{n-1})$$

- ✓ $|\delta_i| \leq u \quad for \quad i = 1, 2, \ldots (n-1)$
- ✓ Relative error in the final computed value:

$$e = \left|\frac{fl(x_1 \times x_2 \times x_3 \times \cdots \times x_n) - (x_1 \times x_2 \times x_3 \times \cdots \times x_n)}{(x_1 \times x_2 \times x_3 \times \cdots \times x_n)}\right|$$
$$= |(1+\delta_1)(1+\delta_2) \ldots (1+\delta_{n-1}) - 1| \leq (1+u)^{n-1} - 1$$

- The quantity $\{(1+u)^{n-1} - 1\}$ may be approximated as 1.06(n-1)u for (n-1)u < 0.1

- ✓ Example: Using a machine with 4-decimal place precision, $t = 4$, $u = 0.5 \times 10^{1-4} = 0.5 \times 10^{-3}$
- ✓ Compute $y = \sqrt{1 + \sin x} - 1$ for $x = 1°$
  - ✓ $fl(x) = \frac{\pi}{180} = 0.1745 \times 10^{-1}$
  - ✓ $fl(\sin x) = 0.1745 \times 10^{-1}$
  - ✓ $fl(1 + \sin x) = 0.1017 \times 10^{1}$
  - ✓ $fl\left(\sqrt{1 + \sin x}\right) = 0.1008 \times 10^{1}$
  - ✓ $fl\left(\sqrt{1 + \sin x} - 1\right) = 0.8000 \times 10^{-2}$
- ✓ True value of y using infinite precision is 0.8688×10^{-2}
- ✓ This is equivalent to performing the same computation in a machine with infinite precision with a starting $x = 0.1606 \times 10^{-1}$
- ✓ Required relative change in x is $(0.1745-0.1606)/0.1745 = 0.7966 \times 10^{-1}$
- ✓ Condition number of the algorithm is $C_a = 0.7966 \times 10^{-1}/0.5 \times 10^{-3} \approx 160$
- ✓ If the algorithm is changed to $y = \frac{\sin x}{\sqrt{1+\sin x}+1}$, the same machine with 4-decimal place precision computes $y = 0.8690 \times 10^{-2}$ with $C_a = 0.4$

Total Relative Error in the output:

Recall: $C_p = \dfrac{Relative\ Error\ in\ the\ output}{Relative\ Error\ in\ the\ input}$

Relative Error in the Output = $C_p \times$ Relative Error in the Input

Relative Error in the Input has two components:

- ✓User specified relative error in the input data = r (say)
- ✓Error introduced due to truncation and round-off errors in the algorithm. Equivalent relative error in the input due to this is computed by backward error analysis as $C_a u$
- ✓Total relative error in the Input data = $(r + C_a u)$

Therefore, Relative Error in the Output = $C_p\,(r + C_a u)$