CMPT-439
Numerical Computation

Fall 2020

Estimation of Errors

The problem

- Errors and uncertainty are unavoidable in computations
- Some are human errors while others are computer errors (e.g. due to precision)
- Errors in numerical methods are unavoidable because these methods only approach a solution giving its approximation
- We therefore look at the types of errors and ways of reducing them

Errors and Uncertainties

- There five types of errors in computation:

1. Mistakes
2. Random error
3. Truncation error
4. Round-off error
5. Propagated error

Errors and Uncertainties

➢ Mistakes:

- logical mistakes in algorithms and programs, which lead to incorrect results
- typographical errors entered with program
- running the program using the wrong data etc.

Errors and Uncertainties

- Random errors: these can be caused for example, by random fluctuations in electronics due to for example power surges. The likelihood is rare but there is no control over them

Errors and Uncertainties

- Truncation or approximation errors: these occur from simplifications of mathematics made to approach a solution
- For example, let us consider substitution of an infinite series by a finite series.
- E.g.: $e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!}$

Errors and Uncertainties

- Truncation or approximation errors: these occur from simplifications of mathematics so that the problem may be solved. For example replace of an infinite series by a finite series.

- E.g.: $e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} \approx \sum_{n=0}^{N} \frac{x^n}{n!} = e^x + \zeta(x, N)$

- Where $\zeta(x, N)$ is the total absolute error

Errors and Uncertainties

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} \approx \sum_{n=0}^{N} \frac{x^n}{n!} = e^x + \zeta(x, N)$$

- The error here is basically a truncation error vanishing as N is taken to infinity
- In practice:

➢ For N much larger than x, the error is small

➢ If x and N are close, then the truncation error is large

Estimation of Errors

- Very seldom any given data are exact, since they originate from measurements. Therefore there is usually some error in the input information
- An algorithm itself usually introduces errors as well, e.g., unavoidable round-offs, etc …
- The output information will then contain some error from both of these sources

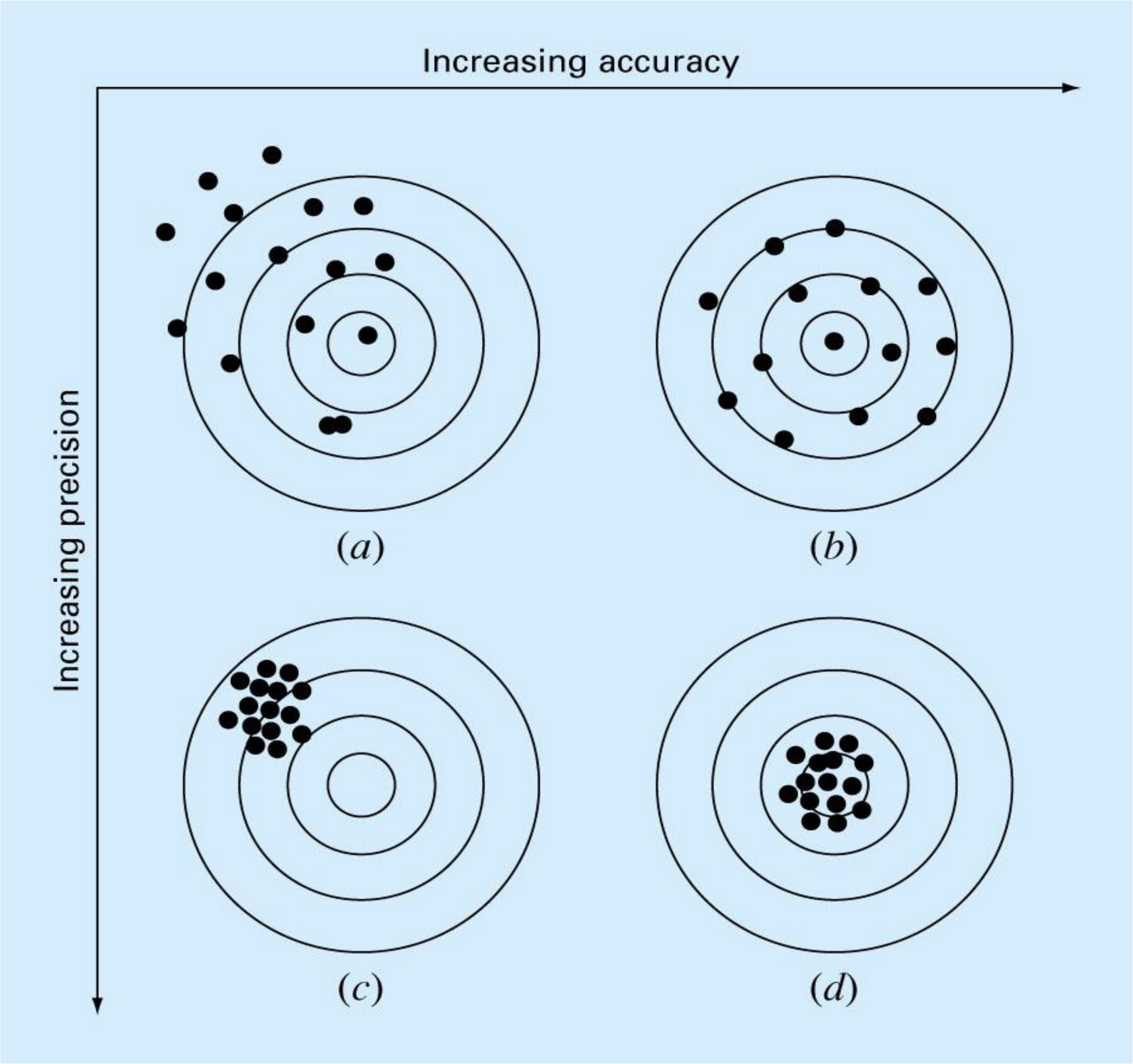
Estimation of Errors

- A common question related to all numerical procedures is how confident we are in the produced results?
- In other words, how much error is present in our calculation and is it tolerable?

Estimation of Errors

- Accuracy. How close is a computed or measured value to the true value
- Precision (or *reproducibility*). How close is a computed or measured value to previously computed or measured values
- Inaccuracy (or *bias*). A systematic deviation from the actual value
- Imprecision (or *uncertainty*). Magnitude of scatter

Accuracy



Significant Figures

- Number of significant digits (figures) indicates precision. Significant digits of a number are those that can be *used* with *confidence,* e.g., the number of certain digits plus one estimated digit.

53,8<u>00</u>   How many significant figures?

5.38×10^4	3
5.380×10^4	4
5.3800×10^4	5

Zeros are sometimes used to locate the decimal point and not significant figures.

0.00001753	4
0.0001753	4
0.001753	4

Estimation of Errors

- Round-off error: since most numbers are represented with imprecision by computers (and general restrictions), this leads to a number being lost.
- The error resulted from rounding or truncation of digits is known as the round-off error.

- E.g.: $2*\left(\frac{1}{3}\right)-\frac{2}{3}=0.6666666-0.6666667=-0.0000001\neq 0$

Error Definitions

- If True Value = Approximate Value + Error, then

e_a = |True value – Approximate Value|

True (Absolute) error

$$\text{Relative error} = \varepsilon_r = \frac{\text{true error}}{\text{true value}}$$

$$\text{Percentage relative error, } \varepsilon_p = \frac{\text{true error}}{\text{true value}} \times 100\%$$

Errors and Uncertainties

- Propagated error: this is an error in later steps of a numerical algorithm due to an earlier error
- This error is added to a local error (e.g. to a round-off error)
- Propagated error is critical as errors can be magnified causing results to be invalid
- The stability of any numerical algorithm depends on how errors are propagated

Calculating the Error: Example

- Find the absolute error for $y = e^x$ if the first 3 terms in the expansion are retained

- Solution: Error = |True value – Approx value|

$$= \left(1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots\right) - \left(1 + x + \frac{x^2}{2!}\right)$$

$$= \frac{x^3}{3!} + \frac{x^4}{4!} + \frac{x^5}{5!} + \ldots$$

Approx value

Calculating the Error

- Absolute error:

  e_a = |True value – Approximate value|

$$e_a = \left|X - X'\right| = \left|Error\right|$$

Calculating the Error

- Absolute error:

  e_a = |True value – Approximate value|

$$e_a = |X - X'| = |Error|$$

- Relative error is defined as:

$$e_r = \left|\frac{Error}{TrueValue}\right| = \left|\frac{X - X'}{X}\right|$$

Calculating the Error

- Percentage error is defined as:

$$e_p = e_r \times 100\% = \left| \frac{X - X'}{X} \right| \times 100\%$$

Calculating the Error: Example

- Suppose 1.414 is used as an approx to $\sqrt{2}$.
- Find the absolute, relative and percentage errors.

Calculating the Error: Example

- Suppose 1.414 is used as an approx to $\sqrt{2}$.
- Find the absolute, relative and percentage errors.
- $\sqrt{2} = 1.41421356$

Calculating the Error: Example

- Suppose 1.414 is used as an approx to $\sqrt{2}$.
- Find the absolute, relative and percentage errors.
- $\sqrt{2} = 1.41421356$

$$e_a = \left|\text{True value} - \text{Approximate value}\right| \quad (\textit{absolute error})$$

$$e_a = |1.41421356 - 1.414|$$

$$= 0.00021356$$

Calculating the Error: Example

- Suppose 1.414 is used as an approx to $\sqrt{2}$.
- Find the absolute, relative and percentage errors.
- $\sqrt{2} = 1.41421356$

$$e_r = \left|\frac{Error}{TrueValue}\right| \qquad (relative\ error)$$

$$e_r = \left|\frac{0.00021356}{\sqrt{2}}\right| \qquad = 0.151 \times 10^{-3}$$

Calculating the Error: Example

- Suppose 1.414 is used as an approx to $\sqrt{2}$.
- Find the absolute, relative and percentage errors.
- $$\sqrt{2} = 1.41421356$$

$$e_r = \left| \frac{Error}{TrueValue} \right| \qquad (relative\ error)$$

$$e_r = \left| \frac{0.00021356}{\sqrt{2}} \right| \qquad = 0.151 \times 10^{-3}$$

$$e_p = e_r \times 100 \qquad = 0.151 \times 10^{-1}\,\% \qquad (percentage\ error)$$

Calculating the Error: Iterative Processes

- Most of numerical methods are iterative.
- Moreover, in the most of real world applications, we usually do not know a true solution a priori. So we have to use
- Approximate Error

$$\varepsilon_a = \left|\text{Current approximation - Previous approximation}\right|$$

Calculating the Error: Iterative Processes

- Most of numerical methods are iterative. In such a case, the errors should be estimated as follows:
- *Iterative approach*

$$\varepsilon_r = \frac{\text{Approximate error}}{\text{Approximation}}$$

$$\varepsilon_a$$

$$\varepsilon_p = \frac{\left|\text{Current approximation - Previous approximation}\right|}{\text{Current approximation}} \times 100\%$$

Mean Square Error and Root Mean Square Error

- The mean squared error (MSE) measures the average of the squares of the errors or deviations—that is, the difference between the estimator and what is estimated
- The root-mean-square error (RMSE) is a measure of the differences between values predicted by a model or an estimator and the values actually observed. The RMSE represents the sample standard deviation of the differences between predicted values and observed values
- Both MSE and RMSE are often used to estimate an error in iterative processes resulted in vectors or matrices. They are also used in signal processing, for example, to evaluate the quality of filtering

Mean Square Error and Root Mean Square Error

- Let $T=(t_1,\dots,t_n)$ be a true vector, $Y=(y_1,\dots,y_n)$ be its approximation. Then
- The mean square error (MSE)

$$MSE=\frac{\sum_{i=1}^{n}(t_i-y_i)^2}{n}$$

- The root-mean-square error (RMSE)

$$RMSE=\sqrt{MSE}=\sqrt{\frac{\sum_{i=1}^{n}(t_i-y_i)^2}{n}}$$

Calculating the Error: Iterative Processes

- Iterative computations are usually repeated until some stopping criterion is satisfied.

$$\varepsilon_a < \varepsilon_s$$

Pre-determined **tolerance** based on the knowledge of your solution

- If the following criterion is met

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

you can be sure that the result is correct to at least *n* significant digits.

Round-off Errors

- Numbers such as π, e, or $\sqrt{7}$ cannot be expressed by a fixed number of significant figures.
- Computers use a base-2 representation, they cannot precisely represent certain exact base-10 numbers
- Fractional quantities are typically represented in computer using a "floating point" form, e.g.,

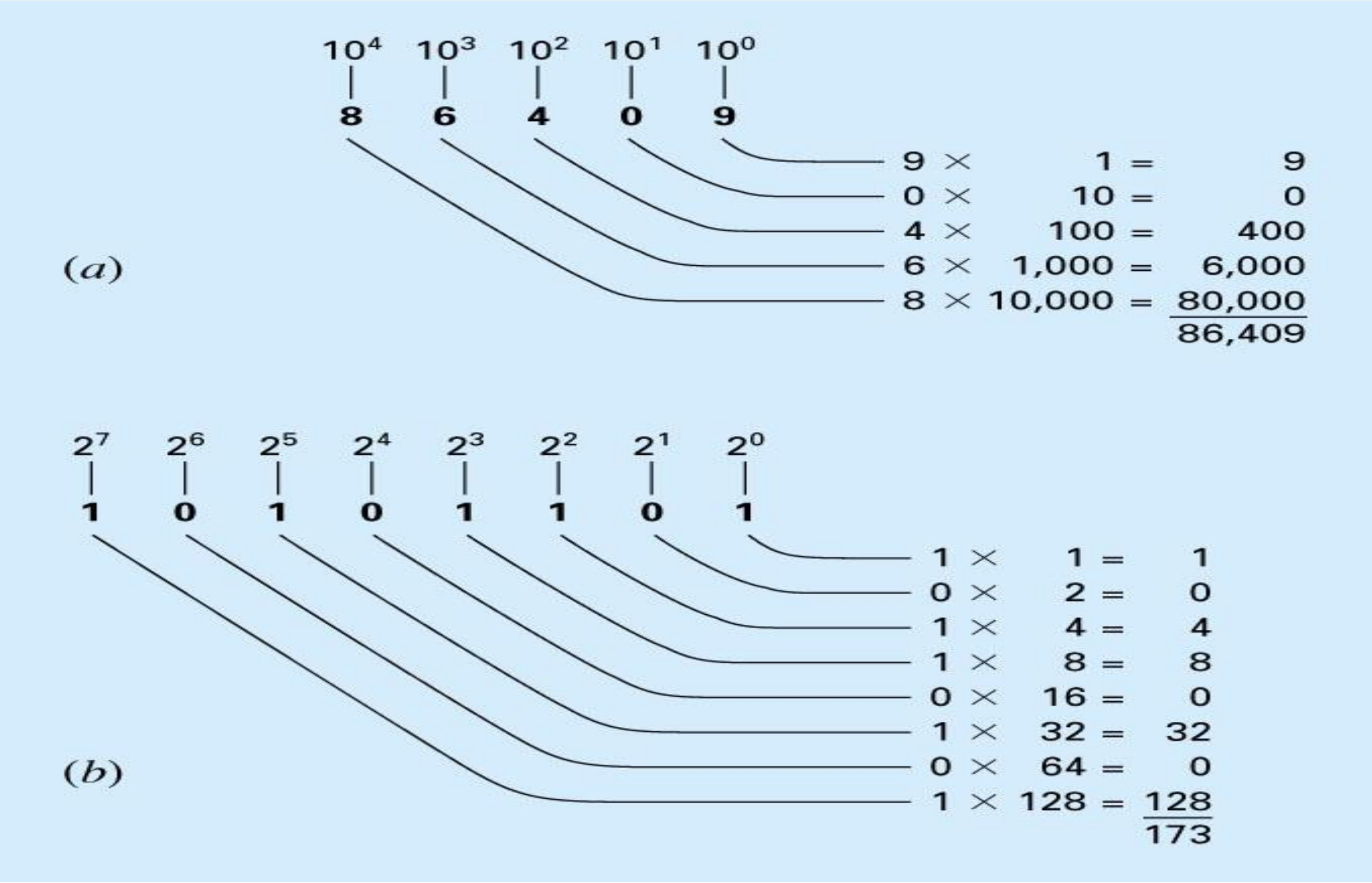
$$m \times b^{e}$$

exponent

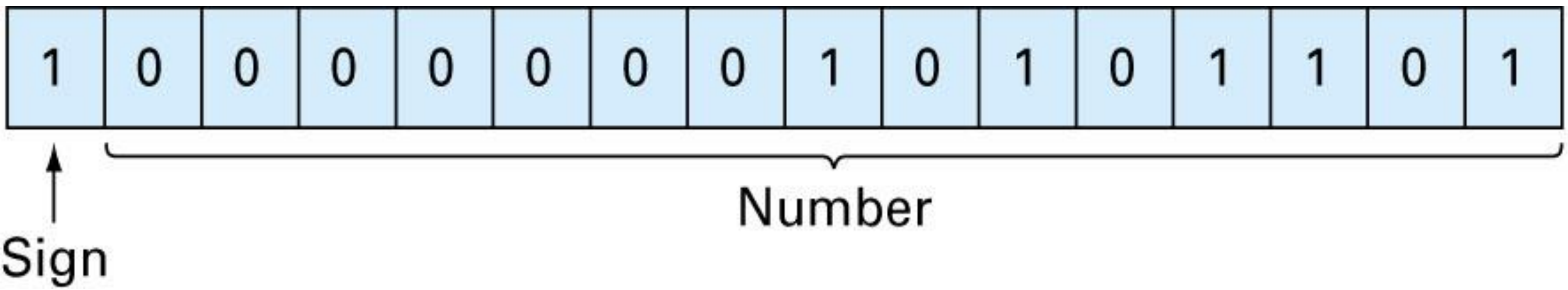
mantissa

Base of the number system used

Representation of Numbers



Representation of Numbers (Integers and fix-point)



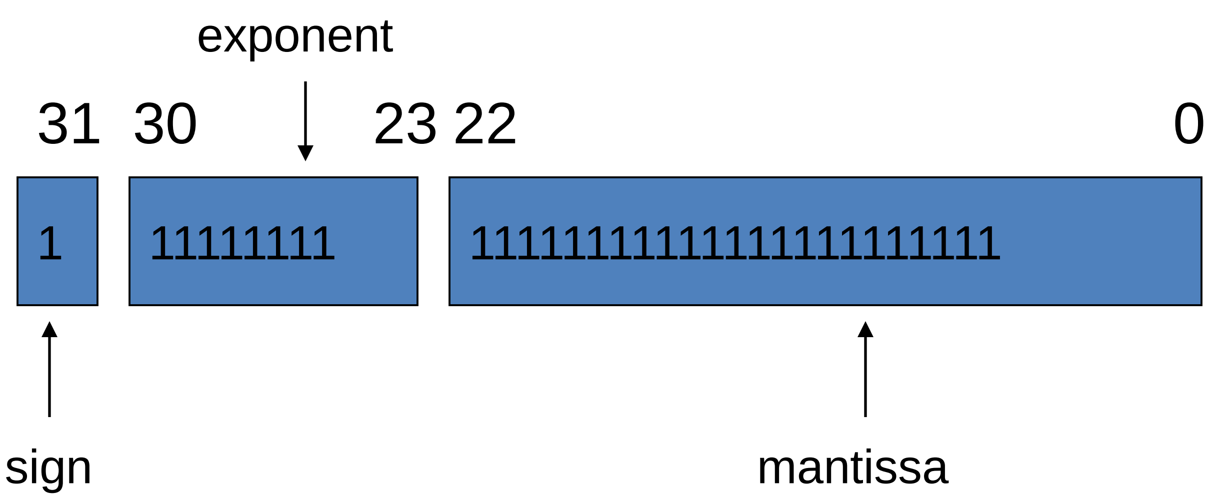
Floating point Representation

- Most computers use the IEEE representation where the floating point number is normalized.
- The two most common IEEE rep are:

1. IEEE Short Real (Single Precision): 32 bits – 1 for the sign, 8 for exponent and 23 mantissa
2. IEEE Long Real (Double Prec): 64 bits – 1 sign bit, 11 for exp and 52 for the mantissa

Representation of Numbers (exponential and floating-point)

- Example of a single precision number.



Floating point Representation

- For example 0.5 is represented as:

0	01111111	10000000000000000000000

- Where the bias is $0111\ 1111_2 = 127_{10}$

Representation of Numbers

156.78 ⏩ 0.15678×10^3 in a floating point base-10 system

$$\frac{1}{34} = 0.029411765$$

Suppose only 4 decimal figures to be stored

$$0.0294 \times 10^0 \qquad 0.1 \le |m| < 1$$

Mantissa

- Normalized to remove the leading zeroes. Multiply the mantissa by 10 and lower the exponent by 1

$0.294\underline{1} \times 10^{-1}$

Additional significant figure is retained

Representation of Numbers

Therefore

for a base-10 system $0.1 \leq m < 1$

for a base-2 system $0.5 \leq m < 1$ Mantissa

- Floating point representation allows both fractions and very large numbers to be expressed in the computer. However,
  - Floating point numbers take up more room
  - Take longer to process than integer numbers
  - Round-off errors are introduced because mantissa holds only a finite number of significant figures

Chopping

- Cutting of digits starting from some position is called Chopping

Example:

π=3.14159265358 to be stored on a base-10 system carrying 7 significant digits.

π =3.141592 chopping error e_t=0.00000065

If rounded

π =3.141593 e_t=0.00000035

- Some machines use chopping, because rounding adds to the computational overhead. Since number of significant figures is large enough, resulting chopping error is negligible.

WELL POSED(CONDITIONED) VS ILL POSED PROBLEMS

Conditioning of a problem and stability

- The accuracy of a solution depends on how a problem is stated (as well as the computer's accuracy).
- Not all solutions of a problem are well posed and hence stable.
- A problem is well posed if a solution (a) exits, (b) is unique, and (c) varies continuously as its parameters vary continuously

Conditioning of a problem and stability

- If the problem is ill-posed it should be replaced by alternative form or another that has a solution which is close enough.
- For example simplifying complicated functions having values which are almost the same.