Structured Programming & an Introduction to Error

Lecture Objectives

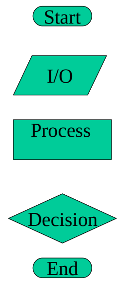
- Review the basic good habits of programming
- To understand basic concepts of error and error estimation as it applies to Numerical Methods

Structured Programming – Ch.2

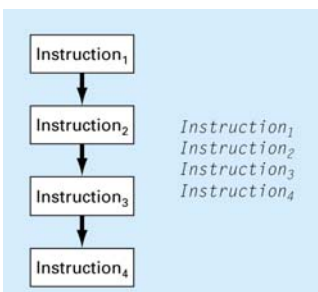
Set of rules that prescribe good habits for a programmer

- Numerical Algorithms are typically composed of 3 types of controls structures:
  1. Sequence
  2. Selection
  3. Repetition
- Tools Use to develop algorithms and visualize before actually writing code
  - Flowcharts – graphical method
  - Pseudocode – simplified computer code statements

Basic Flow Chart Features

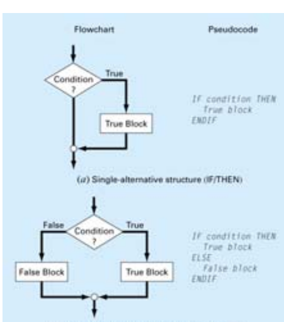


Sequence



(*a*) Flowchart (*b*) Pseudocode

Selection

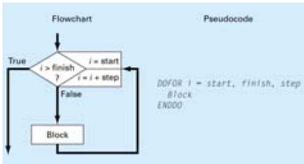


(*a*) Single-alternative structure (IF/THEN)

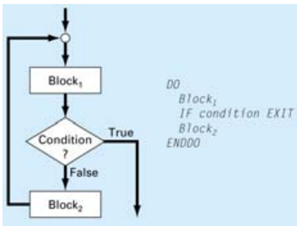
(*b*) Double-alternative structure (IF/THEN/ELSE)

Repetition

Counting Loop



Conditional Loop



Modular Programming

- Breaking up tasks into digestible parts
- The parts should be as independent & self-contained as possible (*Reusable Chunks*)
  - In C/FORTRAN – judicious use of subroutines
  - Matlab – scripts and function

  - Numerical Recipes

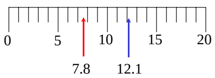
Example

Error & Error Estimation

If we have an analytic solution we can get an exact error, if not we must estimate the error associated with our numerical method

Significant Figures – Confidence in using a number

#Sig digits = Certain digits + 1 Estimated digit



Zeros – When are they significant? (Depends on where they are)

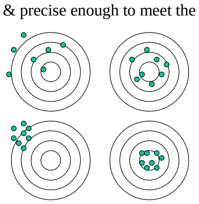
- Preceding zeros:
  - 0.00378 3 Significant Digits
  - 0.0004016 4 Significant Digits
- Following zeros: We use scientific Notation
  - 56,000 56.0×10^3 3 Significant Digits

Accuracy & Precision

(**characterizes error associated with both calculations and measurements)**

- Accuracy: How close is the computed or measured value to the truth?
- Valid: supported by objective truth.
- Precision: How closely do the individually computed or measured values agree with one another?
- Reliable: Produces the same result on successive trials.

* Numerical Methods should be accurate & precise enough to meet the needs of the engineering design problem.



Errors in Numerical Methods

1. Truncation Errors – Results from an approximation of an exact mathematical procedure.

Example: Only using a finite number of terms in an infinite series expansion – Binomial Expansion

$$(1+x)^{\alpha} = 1+\alpha x+\frac{\alpha(\alpha-1)}{2!}x^2+\frac{\alpha(\alpha-1)(\alpha-2)}{3!}x^3+\cancel{HOT}$$

$$\alpha \neq 0, real$$

$$|x|<1$$

2. Round-Off Errors – Results from having numbers with limited significant figures represent exact numbers.

For Both Types of Errors:
True Value = Approximation + Error

Error Definitions

True Error = True Value – Approximation

$$E_t = T - A$$

Normalized True Error – Relative Error (ε_t)

$$\varepsilon_t = \frac{E_t}{T}\times 100\% \longrightarrow \text{True Percent Error}$$

Approximate Error – (E_a) We need to approximate the error when we do not have the "true" value available.

Approximate Relative Error (ε_a)

$$\varepsilon_a = \frac{E_a}{A}\times 100\%$$

How do we find E_a ?
Example: Iterative Approach: $\varepsilon_a = \frac{A_i - A_{i-1}}{A_i}\times 100\%$

Magnitude of Error

Generally, we will compute until the absolute value of the relative error reaches some specified value,

$$|\varepsilon_a| < \varepsilon_s$$

How do we determine ε_s ? To obtain a result that is accurate to at least N significant Figures we can use the following formula:

$$\varepsilon_s = \left(0.5\times 10^{2-N}\right)$$

Round Off Errors

Computers retain only a fixed number of significant figures during a calculation

$\Pi = 3.14159265...$

$e = 2.7183…$

Computers use base-2 representation & can not precisely represent all base-10 numbers

- Word – "fundamental unit of information storage"
  - Consist on binary digits or bits
  - Numbers are stored in 1 or more words
- Base 10 system – 0,1,2,3,4,5,6,7,8,9
  - Example 123,431 – $(1x10^5)+(2x10^4)+(3x10^3)+(4x10^2)+(3x10^1)+(1x10^0)$
- Binary/base 2 system – 0,1
  - Example 1101 – $(1x2^3)+(1x2^2)+(0x2^1)+(1x2^0) = 13$

Integer Representation

Signed binary numbers or *signed magnitude method*:
S = 1 if negative
S = 0 if positive

S															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Range of integers for 16-bit example:
$2^{15}-1$ to -2^{15}

Integer Representation – 5 bit example

1	1	0	0	1
4	3	2	1	0

$$= -1\left[\left(1\times2^3\right)+\left(0\times2^2\right)+\left(0\times2^1\right)+\left(1\times2^0\right)\right]$$

$$= -9$$

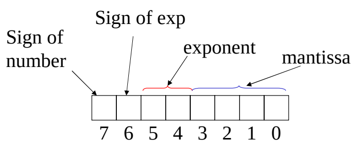
Real or Floating-Point Representation:
Contains a Fractional and Integer part

Mantissa
Fractional part

exponent

$$N = m \cdot b^{e}$$

Base of number system (10)



Floating-Point Representation Example

$$1/27 = 0.037037037\ldots$$

Using 4 digits could be stored as:

$$0.0370 \times 10^{0}$$

Normalize

$$0.3703 \times 10^{-1}$$

Normalizing limits the range of mantissa to:

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

Floating-Point

- Allows us to handle very large and small numbers

but …

DISADVANTAGES:
1. More storage is required than for integers
2. Longer processing time
3. Round-off error is introduced since the mantissa holds a finite number of digits.

Round-Off Error Characteristics

1. Limit to the size (Large & Small) that can be Represented ($10^{-38} < x < 10^{38}$)
2. There are finite number of quantities that can be represented within a range. As a result:
   - Precision is limited
   - Irrational Numbers are not exact
   - Rational #s may not be represented by one of the possible values in the set available on the computer

Quantized Errors – The result of chopping or Rounding
Ex: Π = 3.14159265... If we only have 8 digits:

3.1415926	Chopping	Rounding Reduces error
3.1415927	Rounding	

Round-Off Error Characteristics

3. Interval between numbers (Δx) increases as x increases.
   Example: 4 digit mantissa
   0.4356 x $10^4 \rightarrow \Delta x = 1$
   0.4356 x $10^0 \rightarrow \Delta x = 0.0001$

For Chopping:

$$\frac{|\Delta x|}{|x|} \le \varepsilon_m$$

ε_m is the "machine epsilon" – sets the bounds for the relative Quantized error.

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

Sig digits in mantissa

Number base

When & why would we be Interested in ε_m ?

Extended Precision

Single Precision – For most engineering application o.k. Typically, 7 significant base-10 digits for 24 bit mantissa
→Range 10^{-39} to 10^{-38}

Double Precision - 15 to 16 base-10 digits → Range 10^{-308} to 10^{-308}

- Round-Off Error is mitigated
- Computational time increases

Arithmetic Manipulation Errors

- Simple operation → R.O. Error
- Consider computer w/4 digit mantissa

1. Addition: Add 2.345 & 0.0123

$$\begin{array}{r} 0.2345\times10^{1} \\ +\underline{0.1234\times10^{-1}} \end{array} \xrightarrow{\text{Modify smaller exponent to match larger}} \begin{array}{r} 0.2345\times10^{1} \\ +\underline{0.001234\times10^{1}} \\ 0.235734\times10^{1} \end{array}$$

$$\downarrow \text{Chop}$$

Last 2 digits have been lost!

$$\varepsilon_t = \frac{2.35734-2.357}{2.35734}\times100\% = 0.014\% \qquad 0.2357\times10^{1}$$

Arithmetic Manipulation Errors

- Simple operation → R.O. Error
- Consider computer w/4 digit mantissa

2. Subtraction: What if we have 2 nearly equal numbers? Subtract 12.33 from 12.34

$$\begin{array}{r} 0.1234\times10^{2} \\ -\underline{0.1233\times10^{2}} \\ 0.0001\times10^{2} \end{array} \xrightarrow{\text{Normalize}} 0.1\underbrace{000}_{\text{3 non-significant zeros added!}}\times10^{-1}$$

Subtractive cancellation:
One of the most troublesome
Round off errors.

Arithmetic Manipulation Errors

- Simple operation → R.O. Error
- Consider computer w/4 digit mantissa

3. Multiplication: Exponents are added & Mantissas multiplied

$$\left(0.1524\times10^{2}\right)\cdot\left(0.5924\times10^{2}\right) = 0.09028176\times10^{4}$$

$$\downarrow \text{Normalize}$$

$$0.9028176\times10^{3}$$

$$\downarrow \text{Chop}$$

$$0.9028\times10^{3}$$

4. Division: Mantissas are divided and exponents subtracted.

Arithmetic Manipulation Errors

5. Large Numbers of Computations: Cumulative effect of many small round-off errors.

See Example 3.6 in Text for a FORTRAN Example:

- Matlab uses double precision for all calculations

6. Adding Large and Small Numbers:

$$\begin{array}{r} 20000 \\ +\ \ 0.1 \\ \hline \end{array} \xrightarrow{\text{Normalize}} \begin{array}{r} 0.20000\ \times 10^{5} \\ +\ 0.000001\ \times 10^{5} \\ \hline 0.200001\ \times 10^{5} \end{array} \xrightarrow{\text{Chop}} 0.2000\ \times 10^{5}$$

0.1 term is entirely lost!

7. Smearing – Summation of Series
   - Occurs whenever individual terms of a series are larger the the total summation

$$e^{x} = 1 + x + \frac{x^{2}}{2} + \frac{x^{3}}{3!} + ...$$

x >0 No problem
x <0 sign switching can occur