Numerical Analysis Basics

Larry Caretto
Mechanical Engineering 309
Numerical Analysis of Engineering Systems

February 5, 2014

California State University Northridge

Outline

- Programming Languages
- Binary Numbers and Data Types
  - Limits on the size and accuracy of numbers stored on computers
- Errors in computing
- Character and date representations
- Introduction to numerical solution of algebraic equations

Programming Languages

- Instructions to manipulate information stored in computer memory
- Machine language that actually does work is all in binary numbers
- Assembly language, used in some applications, has commands that are similar to actual computer operation
- Higher-level languages are easier for humans to understand and use

Some Higher-Level Languages

- Fortran – earliest higher-level language
- COBOL – original business language
- PASCAL – used for instruction in 1970s
- ADA – developed by US Department of Defense in 1970s
- C – developed at Bell labs to be applied as higher and lower level language
- C++ – variant of C that uses object-oriented programming

BASIC and VBA

- **B**eginners **A**ll-purpose **S**ymbolic **I**nstruction **C**ode developed for instructional use at Dartmouth in 1964
- Used by various microcomputers in 1970s (Apple, Microsoft, *etc*.)
- **V**isual **B**asic for **A**pplications is object-oriented version used to support many programs (Office, LabView, SolidWorks, *etc*.

MATLAB

- **MAT**rix **LAB**oratory – initially developed as interface for students to Fortran linear algebra programs
- Widely used in control system analysis
- Provides simple interface for quick answers to complex problems
- MATLAB coding language available for advanced problem solving

Common Language Features

- Data types: integer, real, string, …
- Replacement statements: assign a value to a variable
- Expressions: mathematical, logical, *etc*.
- Choice statements: if, case
- Looping statements
  - Count controlled loops
  - Conditional loops
- Arrays

California State University Northridge 7

A Binary Machine

- Computers use binary numbers
  - 0, 1, 10, 11, 100, 101, 110, 111, 1000, …
- This places limits on accuracy
- Data types translate zeros and ones in computer hardware into data (numbers and text) that humans recognize
- The number of binary bits (a binary digit, 0 or 1) used to represent a number affects its maximum size and accuracy

What are the next numbers in this sequence?

1001, 1010, 1011, 1100, 1101, 1110

1111
10000
10001
10010
10011

California State University Northridge 8

Numbers and Bases

- We want to display binary numbers and show their equivalence to normal (base-ten) numbers
- We will also mention octal (base eight) and hexadecimal numbers (base 16)
- To discuss this we will develop a general format starting with base-ten numbers using an example to get a general format

California State University Northridge 9

Decimal Numbers

- What does the decimal number 132 mean?
  - It's 2 times 1 + 3 times 10 + 1 times 100
  - It's 2 times 10^0 + 3 times 10^1 + 1 times 10^2
- How can we generalize this?
  - Define d_k as the digit in position k of a number, counting from right to left, with d_0 being the rightmost digit
  - In the example of 132, $d_0 = 2$, $d_1 = 3$ and $d_2 = 1$

What are d_k values for 7491?

$d_0 = 1$, $d_1 = 9$, $d_2 = 4$ and $d_3 = 7$

California State University Northridge 10

General Decimal Numbers

- Start with example from previous chart
  - We wrote 132 as 2 times 10^0 + 3 times 10^1 + 1 times 10^2
  - So $132 = 2(10^0) + 3(10^1) + 1(10^2)$
  - We also said that for 132 , $d_0 = 2$, $d_1 = 3$ and $d_2 = 1$
  - So, for the general three-digit number we have $d_2d_1d_0 = d_2(10^2) + d_1(10^1) + d_0(10^0)$
- How can we write an N-digit number?

$$d_{N-1}d_{N-2}\ldots\ldots d_1d_0 = d_{N-1}(10^{N-1}) + d_{N-2}(10^{N-2})\ldots\ldots + d_1(10^1) + d_0(10^0)$$

California State University Northridge $d_{N-1}d_{N-2}\ldots\ldots d_1d_0 = \sum_{k=0}^{N-1} d_k(10^k)$ 11

Generalizing the Base

- Rewrite general base-10 equation

$$d_{N-1}d_{N-2}\ldots\ldots d_1d_0 = \sum_{k=0}^{N-1} d_k(10^k)$$

Can use base, b, as subscript for clarity

- to apply to any base, b

$$d_{N-1}d_{N-2}\ldots\ldots d_1d_0 = (d_{N-1}d_{N-2}\ldots\ldots d_1d_0)_b = \sum_{k=0}^{N-1} d_k(b^k)$$

- Common bases for computing: 2, 8, 16
  - Base 16 digits use $a_{16} = 10_{10}$, $b_{16} = 11_{10}$, $c_{16} = 12_{10}$, $d_{16} = 13_{10}$, $e_{16} = 14_{10}$, and $f_{16} = 15_{10}$
    - What is abc_{16} in base ten?
    - $(abc)_{16} = 12(16^0) + 11(16^1) + 10(16^2) = 2748_{10}$

California State University Northridge $c_{16} = 12_{10}$ $b_{16} = 11_{10}$ $a_{16} = 10_{10}$ 12

Different Bases

Decimal	0	1	2	3	4	5
Binary	0	1	10	11	100	101
Octal	0	1	2	3	4	5
Hex	0	1	2	3	4	5
Decimal	6	7	8	9	10	11
Binary	110	111	1000	1001	1010	1011
Octal	6	7	10	11	12	13
Hex	6	7	8	9	a	b

- One octal digit ranges over all possible combinations of 3 binary digits (000 to 111)
- One hex digit ranges from 0000 to 1111

California State University Northridge 13

Integer Numbers on Computer

- Integer data types fill memory available with zeros and ones (one bit required)
- What is maximum number that can be stored in 2 bytes (16 bits)?

$$d_{15}d_{14}\ldots\ldots d_1d_0 = \sum_{k=0}^{15} d_k(2^k) \qquad so\ that\ all\ d_k = 1\ gives$$

$$1111111111111111 = \sum_{k=0}^{15} 1(2^k) = 2^{15} + 2^{14} + \ldots\ldots + 2^1 + 2^0 = 65\,535$$

- Can show maximum size for N bits = $2^N - 1$

California State University Northridge 14

Integer Numbers on Computer II

- 16-bit integer numbers can range from 0 to $65{,}535_{10}$
- Use offset to have range from $-32{,}768_{10}$ to $32{,}767_{10}$
- Long data type with 4 bytes (32 bits) has range of $2^{32} - 1 = 4{,}294{,}967{,}295$
- With offset range is $-2{,}147{,}483{,}648_{10}$ to $2{,}147{,}483{,}647_{10}$

California State University Northridge 15

Real Numbers on Computer

- Use power notation for number. For example, -0.123456×10^4
  - In this example 123456 is called the significand and 4 is called the exponent
  - One bit is required for the sign (+ or –)
  - IEEE standard 754 for "floating-point" numbers started 1985, revised 2008
  - Single data type uses 4 bytes (32 bits) with one sign bit, 23 bits for the significand and 8 bits for the exponent

California State University Northridge 16

Real Numbers on Computer

- Use power notation for number. For example, -0.123456×10^4
  - In this example 123456 is called the significand and 4 is called the exponent
  - One bit is required for the sign (+ or –)
  - IEEE standard 754 for "floating-point" numbers started 1985, revised 2008
  - Single data type uses 4 bytes (32 bits) with one sign bit, 23 bits for the significand and 8 bits for the exponent

California State University Northridge 17

Real Numbers on Computer II

  - A binary significand will have the form .1dddd… (d's are remaining digits, 0 or 1)
  - The lead 1 is not stored so the 23 bits for the significand are effectively 24 bits
  - The maximum size of the "24"-bit significand is $2^{24} - 1 = 16{,}277{,}214$; $\log_{10}(16{,}277{,}214) = 7.22_{10}$ significant figures
  - Eight-bit exponent size = $2^8 - 1 = 255_{10}$
    - Scaled from –128 to 127 giving maximum power of $2^{127} = 10^{38.23}$

California State University Northridge 18

Real Numbers on Computer III

- Type double uses 8 bytes (64 bits)
  - 52 bits for significand; 11 for exponent
  - Maximum exponent is 10^{308}
  - Gives about 16 significant figures
- VBA does not have quadruple type that uses 16 bytes (128 bits)
  - 112 bits for significand, 15 for exponent
  - Maximum exponent is 10^{4931} and accuracy is 34 significant figures

California State University Northridge 19

Numerical Analysis Errors

- **Truncation error** caused by converting calculus equations into algebraic equations by truncating Taylor series
- **Roundoff error** caused by inexact conversion of decimal numbers into binary representation
- **Error propagation** is growth in errors as calculations proceed due to different error types

California State University Northridge 20

Engineering Errors

- **Modeling Errors** arise when a simplified model of a complex process is used
  - Done when some parts of process do not have complete mathematical description
  - Also done to reduce computer time and/or storage for detailed processes
- **Human error**, sometimes called blunders

California State University Northridge 21

Character Codes

- ASCII uses 1 byte (0 to $377_8 = 255_{10}$)
  - See char(N) function on Excel worksheet
    - N = 65 to 90 is A to Z
    - N = 97 to 122 is a to z
- Unicode used to accommodate multiple character sets for different languages (Asian, Arabic, Hebrew, *etc*.)
  - Range is 0 to $10FFFF_{16} = 1114111_{10}$
  - Initial range same as ASCII

California State University Northridge 22

Date Variables

- Really date and time
- Excel and VBA use 8 bytes for storage
- Integer part is number of days since reference date
  - Date range is 1/1/100 to 12/31/9999
- Fractional part is time in fraction of a day (9 am = 0.375, noon = 0.5, *etc*.)
  - Use format function to convert this fraction to conventional hours, minutes, seconds

California State University Northridge 23

Solution of Equations

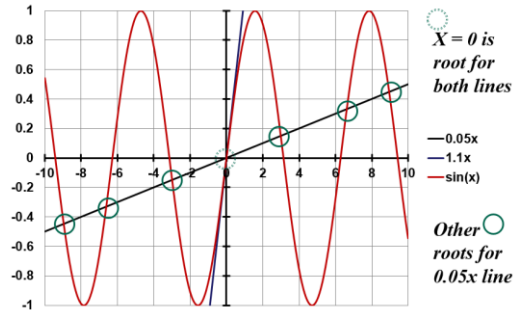
- First topic in numerical analysis
- Some equations like $3x + 15 = 5(4x+20)$ or $0.2 = \sin(x)$ can be solved for x
  - Get $x = -5$ and $x = \sin^{-1}(.2)$ for examples
- Some equations cannot be solved directly, e.g.: $3x^2 = \cos(x)$
- Want **general** methods for solving such equations

California State University Northridge 24

Example Problem

- What is solution of ax = sin(x)
- Could use graphing calculator
- Algorithms for finding roots of equations used as parts of larger computer programs that requires such solutions
- Graphing equation to be solved is used here to show how roots (solutions) can be visualized
  - Note: there may be more than one root

Plot to solve ax = sin(x)



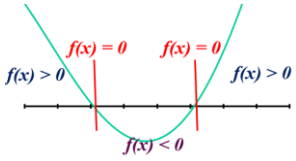
Methods for Finding Roots

- Two classes of methods
  - One initial guess
  - Two initial guesses that "bracket" root (*i.e.* root lies between the initial guesses)
    - Some require more than two initial guesses
- When location of root cannot be estimated a search procedure is required to bracket root
- Methods that bracket root are usually slower but surer

General Algorithms

- Any equation in one unknown cay be written in the form f(x) = 0
- The equations shown previously, ax = sin(x) (a = 1.1 or 0.05), can be written as ax – sin(x) = 0 or sin(x) – ax = 0
- Almost all algorithms based on this form
- Root means that f(x) = 0
- As |f(x)| decreases we are getting closer to a root

Bracketing a Root



Plot shows two roots, f(x) = 0 and values of f(x) on opposite sides of root

- Basic idea: when a root, f(x) = 0, is bracketed the values of f(x) on opposite sides of the root have opposite signs
- If f(x)f(x+Δx) < 0 there is a root, f(x) = 0, between x and x+Δx

How to Bracket a Root

- Often some physical information about a problem will let you know that there is a root at some approximate location x*
- Pick a value of Δx and find f(x* + Δx) and f(x* – Δx)
  - If the product of these two f values is negative, the root is bracketed
  - If product is positive increase Δx and retry
  - If Δx is too large you may have bracketed two roots and have f(x* + Δx)f(x* – Δx) > 0

Process and Notation

- In solving f(x) = 0 we use iteration
- We make an initial estimate, called x_0, of the correct root (where f(x) = 0)
- Sometimes we make two initial estimates, x_0 and x_1 (or even more)
- We have an iteration procedure that uses previous estimates, x_k, x_{k-1}, *etc*. to find a new value x_{k+1}
- Iterations continues until x value is found that gives desired accuracy

California State University Northridge 31

Secant Method

- Starts with two initial guesses, x_0 and x_1, which need not bracket root
- Each iteration uses values of two successive guesses, x_k and x_{k-1} to find new guess, x_{k+1}
- Approximate behavior of f(x) near guess as straight line: $f - f(x_k) = m(x - x_k)$
  - Slope m found from two latest guesses

$$m = \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}$$

California State University Northridge 32

Secant Method II

- Substitute equation for slope, m, into linear approximation: $f - f(x_k) = m(x - x_k)$

$$f - f(x_k) = \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}(x - x_k)$$

- Take new guess, x_{k+1} as value of x that sets f = 0 in linear approximation

$$0 - f(x_k) = \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}}(x_{k+1} - x_k)$$

$$x_{k+1} = x_k - f(x_k)\frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})}$$

California State University Northridge 33

Secant Method III

- Start with two initial guesses x_1 and x_0
- Repeat the following steps to get a new guess, x_{k+1}, from the guesses x_k and x_{k-1}
  - Compute x_{k+1} from the equation

$$x_{k+1} = x_k - f(x_k)\frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})}$$

  - Compute $f(x_{k+1})$
- Repeat the process until two successive values are "close enough"

California State University Northridge 34

Secant Example

- Solve f(x) = 0.05x – sin(x) = 0
- Pick $x_0 = 2$ and $x_1 = 2.5$
  - $f(x_0) = f(2) = 0.05(2) - \sin(2) = -0.809$
  - $f(x_1) = f(2.5) = 0.05(2.5) - \sin(2.5) = -0.473$

Initial guesses do not have to bracket root

- Apply general equation to first iteration

$$x_{k+1} = x_k - f(x_k)\frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})}$$

$$x_2 = x_1 - f(x_1)\frac{x_1 - x_0}{f(x_1) - f(x_0)}$$

$$x_2 = 2.5 - (-0.473)\frac{2.5 - 2}{-0.473 - (-0.809)} = 3.20$$

$$f(x_2) = 0.05(3.20) - sin(3.21) = 0.224$$

California State University Northridge 35

Secant Example II

- Note: All significant figures of calculator or spreadsheet used in calculations
  - Rounding shown here to save space
- Apply general equation to second iteration

$$x_{k+1} = x_k - f(x_k)\frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})}$$

$$x_3 = x_2 - f(x_2)\frac{x_2 - x_1}{f(x_2) - f(x_1)}$$

$$x_3 = 3.20 - (0.224)\frac{3.20 - 2.5}{0.224 - (-0.473)} = 2.98$$

$$f(x_3) = 0.05(2.98) - sin(2.98) = -0.0131$$

Can you find x4 and $f(x_4)$?

California State University Northridge 36

Secant Example III

k	x_k	$f(x_k)$	$\lvert x_k - x_{k-1}\rvert$
0	2	-8.09E-01	
1	2.5	-4.73E-01	5.0E-01
2	3.20493820516827	2.24E-01	7.0E-01
3	2.97884928100731	-1.31E-02	2.3E-01
4	2.99134973893018	-1.11E-04	1.3E-02
5	2.99145653304953	1.04E-07	1.1E-04
6	2.99145643339981	-7.95E-13	1.0E-07
7	2.99145643340058	0.00E+00	7.7E-13
8	2.99145643340058	0.00E+00	0.0E+00

California State University Northridge

37

Quiz One solutions I

```
Function myInt(a as Double, b as _
    Double, n as Double) as Double
   If n <> -1 Then
      myInt = (b^(n+1) – a^(n+1)) _
                              /(n+1)
   ElseIf a*b > 0 Then
      myInt = log(b/a)
   Else
      myInt = "Undefined"
   End If
End Function
```

California State University Northridge

38

Quiz Solutions II

- Write the following Excel formula in cell B4 and copy the equation to cells C4:E4 to get the results for all cells:
  - =myInt(A1, B1, C1,)
- For part 3 the values in the cells give

  $[2^{(1+1)} – 1^{(1+1)}] / (1 + 1) = 1.5$

  $[(-2)^{(1+1)} – 2^{(1+1)}] / (1 + 1) = 0$

  $[(-1)^{(4+1)} – (-1)^{(4+1)}] / (4 + 1) = 0$

California State University Northridge

39