Numerical Analysis Basics

Larry Caretto
Mechanical Engineering 309
Numerical Analysis of Engineering Systems

February 5, 2014

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

- Beginners All-purpose Symbolic Instruction Code developed for instructional use at Dartmouth in 1964
- Used by various microcomputers in 1970s (Apple, Microsoft, etc.)
- Visual Basic for Applications is object-oriented version used to support many programs (Office, LabView, SolidWorks, etc.)

MATLAB

- MATrix LABoratory – 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

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

General Decimal Numbers
- Start with example from previous chart
  - We wrote 132 as 2 times $10^2$ + 3 times $10^1$ + 1 times $10^0$
  - So 132 = $2(10^2) + 3(10^1) + 1(10^0)$
  - 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?
  \[d_{N-1}d_{N-2}...d_1d_0 = \sum_{i=0}^{N-1} d_i(10^i)\]

Generalizing the Base
- Rewrite general base-10 equation
  \[d_{N-1}d_{N-2}...d_1d_0 = \sum_{i=0}^{N-1} d_i(b^i)\]
- to apply to any base, $b$
- 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?
    \[ (abc)_{16} = 12(16^2) + 11(16^1) + 10(16^0) = 2748_{10} \]
**Different Bases**

<table>
<thead>
<tr>
<th>Base</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decimal</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Binary</td>
<td>0</td>
<td>1</td>
<td>10</td>
<td>11</td>
<td>100</td>
<td>101</td>
</tr>
<tr>
<td>Octal</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Hex</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Decimal</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>Binary</td>
<td>6</td>
<td>7</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
</tr>
<tr>
<td>Octal</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>a</td>
<td>b</td>
</tr>
</tbody>
</table>

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

**Integer Numbers on Computer I**

- 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)?

\[ d_1d_0d_{-1}d_{-2} = \sum_{k=-1}^{15} d_k (2^k) \text{ so that all } d_k = 1 \text{ gives} \]

\[ 1111111111111111 \]

- Can show maximum size for N bits = \(2^N - 1\)

**Real Numbers on Computer I**

- Use power notation for number. For example, \(-0.123456 \times 10^4\)

- 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

- A binary significand will have the form .1dd... (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 \(2^{24} - 1 = 16,277,214\); \(\log_{10}(16,277,214) = 7.22\) significant figures
- Eight-bit exponent size \(= 2^8 - 1 = 255\)

- Single data type uses 4 bytes (32 bits) with one sign bit, 23 bits for the significand and 8 bits for the exponent

- Scaled from \(-128\) to \(127\) giving maximum power of \(2^{127} = 10^{38.23}\)
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

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

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

Character Codes

- ASCII uses 1 byte (0 to 377 = 255)
  - 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 = 1114111
  - Initial range same as ASCII

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

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 $x = \sin^{-1}(0.2)$ for examples
- Some equations cannot be solved directly, e.g.: $3x^2 = \cos(x)$
  - Want **general** methods for solving such equations
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

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 can 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 $f(x)f(x+\Delta x) < 0$ there is a root, $f(x) = 0$, between $x$ and $x+\Delta x$

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 $\Delta x$ and find $f(x^* + \Delta x)$ and $f(x^* - \Delta x)$
  - If the product of these two $f$ values is negative, the root is bracketed
  - If product is positive increase $\Delta x$ and retry
  - If $\Delta x$ is too large you may have bracketed two roots and have $f(x^* + \Delta x)f(x^* - \Delta x) > 0$
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 continue until \( x \) value is found that gives desired accuracy

**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: \( f(x_k) = m(x - x_k) \)
  - Slope \( m \) found from two latest guesses
  \[ m = \frac{f(x_k) - f(x_{k-1})}{x_k - x_{k-1}} \]

**Secant Method II**
- Substitute equation for slope, \( m \), into linear approximation: \( f - f(x_k) = m(x - x_k) \)
  \[ f - f(x_k) = f(x_k) - f(x_{k-1})(x - x_k) \]
- Take new guess, \( x_{k+1} \) as value of \( x \) that sets \( f = 0 \) in linear approximation
  \[ 0 - f(x_k) = f(x_k) - f(x_{k-1}) \frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})} \]
  \[ x_{k+1} = x_k - f(x_k) \frac{x_k - x_{k-1}}{f(x_k) - f(x_{k-1})} \]

**Secant Example**
- Solve \( f(x) = 0.05x - \sin(x) = 0 \) initial guesses do not have to bracket root
- Pick \( x_0 = 2 \) and \( x_1 = 2.5 \)
  \[ -f(x_0) = f(2) = 0.05(2) - \sin(2) = -0.809 \]
  \[ -f(x_1) = f(2.5) = 0.05(2.5) - \sin(2.5) = -0.473 \]
- Apply general equation to first iteration
  \[ x_2 = 2.5 - \frac{0.473}{-0.809} = 3.20 \]
  \[ f(x_2) = 0.05(3.20) - \sin(3.21) = 0.224, 0.25 \]
### Secant Example III

| k | $x_k$ | $f(x_k)$ | $|x_k - x_{k-1}|$ |
|---|---|---|---|
| 0 | 2 | -8.09E-01 | |
| 1 | 2.5 | -4.73E-01 | 5.0E-01 |
| 2 | 3.20493820516827 | 2.24E-01 | 7.0E-01 |
| 3 | 2.97884973893018 | -1.11E-02 | 1.3E-01 |
| 4 | 2.99134973893018 | 1.04E-07 | 1.1E-04 |
| 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 |

### Quiz One solutions I

Function `myInt(a as Double, b as Double, n as Double) as Double`

If $n <> -1$ Then

```vba
myInt = (b^(n+1) - a^(n+1)) / (n+1)
```

ElseIf $a*b > 0$ Then

```vba
myInt = log(b/a)
```

Else

```vba
myInt = “Undefined”
```

End If

End Function

### 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:
  ```excel
  =myInt(A1, B1, C1)
  ```
- For part 3 the values in the cells give
  
  
  \[
  \frac{2^{(1+1)} - 1^{(1+1)}}{(1 + 1)} = 1.5
  \]
  
  \[
  \frac{[(-2)^{(1+1)} - 2^{(1+1)}}{(1 + 1)} = 0
  \]
  
  \[
  \frac{[(-1)^{(4+1)} - (-1)^{(4+1)}}{(4 + 1)} = 0
  \]