CS 221 Lecture

Tuesday, 6 September 2011

Today’s Agenda

1. Announcements
2. Using MATLAB
3. Binary Representation of Values
4. Interlude on Precision and Accuracy
5. Variables, Types, and Expressions
6. Recap
7. Lab 2 Preview

1. Announcements

- Dates of in-class quizzes:
  - Tuesday 4 October
  - Tuesday 25 October
  - Tuesday 29 November
- Homework Assignment 1 will be available on web site tomorrow
  - Due Wednesday 21 September
  - Turn in via CS portal (same as labs)
- Lab 1 is due Thursday
- Remember:

  Bring your textbook to lab!

2. Using MATLAB

- The “Desktop” is made up of several windows
  - Command Window
  - Command History
  - Workspace Window
  - Current Folder
  - Start Menu
  - Window Menus
- Each of these can be closed, “undocked”, maximized

Using MATLAB

- Working in the Command Window
  - Type an expression, MATLAB evaluates it, prints result (just like a TI-83)
  - Controlling the output in the command window
    - format command (“format short”, “format long”)
    - semicolon suppresses echoing

Using MATLAB

- Other Windows:
  - Workspace Window

    Keeps track of each named value you create

    - Type (“Class”), size, value
  - Current Folder Window

    Shows user files (containing programs or data) that MATLAB knows about

    - Standard navigation controls
- Reorganizing your Desktop
  - Docking/undocking, min-/maximizing windows

3. Binary Representation of Values

- Digital electronics deal with 1's and 0's (only)
  - Calculators
  - Phones
  - PCs
  - Supercomputers
  - DVD players
  - Music players
- Computers store and compute with values as groups of bits

  (= "binary digits")

Binary and Decimal Representation

- A car's odometer represents a value as a sequence of decimal digits:

10^5 place

0 1 6 7 9 2

ones place ($=10^0$)

thousands place ($=10^3$)

tens place ($=10^1$)

- Inside a computer, a value is represented as a sequence of binary digits:

0 0 0 0 0 1 0 0 0 0 0 1 1 0 0 1 1 0 0 0

ones place ($=2^0$)

2^{14} place

256s place ($=2^8$)

fours place ($=2^2$)

Bits and Bytes

- A byte is a group of 8 bits
  - The bits in a byte are ordered, like the digits in a number
  - What's the biggest integer that can be stored in one byte? It depends.
  - If we only want to encode unsigned values (i.e., no negative numbers): 255 (=$2^8 - 1$)
  - If we need to encode signed (both negative and positive) values: 127 (=$2^7 - 1$)
  - There are 256 (=2^8) different bytes
- Computers generally represent values using fixed-size groups of bytes – 2, 4, or 8 bytes
  - i.e., 16, 32, or 64 bits

Binary Representation of Values
(cont.)

- Obviously only finitely many distinct values can be represented with a fixed number of bits
  - 8 bits: 2^8 = 256 values
  - 16 bits: 2^{16} = 65,536 values
  - 32 bits: 2^{32} = 4,294,967,296 values
  - 64 bits: 2^{64} = 18,446,744,073,709,551,616 values
- Neither Excel nor MATLAB can represent arbitrarily large or arbitrarily small numbers

Note: some tools can do arbitrary-precision calculations

...but they are relatively slow

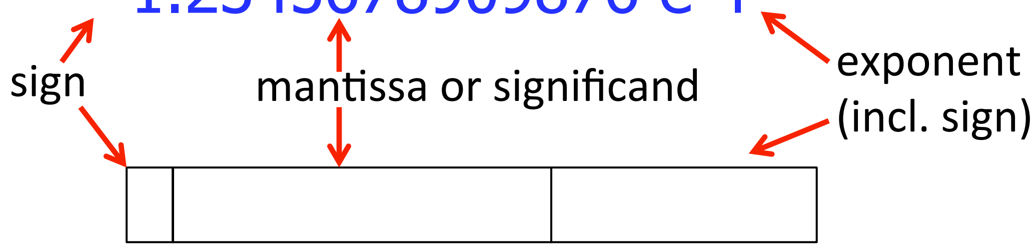
Binary Representation of Values
(cont.)

exponent

mantissa

- What about "real numbers"?
  - E.g., 3.141592653589793..., 12345.678909876
- Represented as fixed-size floating-point binary numbers
  - "single-precision:" 32 bits
  - "double-precision:" 64 bits
- Similar to (decimal) scientific notation:

1.2345678909876 e 4

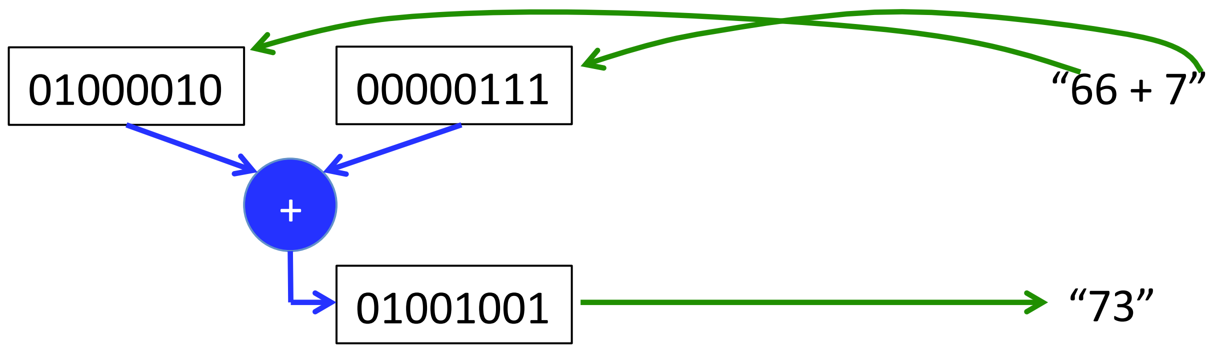


Floating-Point Values

- Single-precision (4 bytes)
  - 1-bit sign
  - 8-bit exponent (power of ten)
  - 23-bit significand (binary fraction)
  - Effect is 7+ decimal digits of precision
- Double-precision (8 bytes)
  - 1-bit sign
  - 11-bit exponent (power of ten)
  - 52-bit significand (binary fraction)
  - Effect is just under 16 decimal digits of precision

Internal Representation vs. External

- Note well: the internal, binary representation is not the same as the external, displayed (decimal) representation



- Both Excel and MATLAB allow the user to control the way numbers are displayed
  - E.g., number of decimal places displayed
  - Internal values are rounded to the displayed precision

Decimal-to-Binary Conversion

- Write down powers of 2 until you get one larger than the number to be converted

  (To get powers of 2: start with 1 and keep doubling)
- Subtract the next-to-last power of 2 from the current number and write down a 1
- Repeat until the current number is zero:
  - Go to the next lower power of 2:
    - If it is larger than the current number, write a 0 (to the right of the previous digits)
    - Otherwise, write a 1 (to the right of the previous digits) and subtract it from the remaining number

Example: Convert 333 to binary

Find the leftmost "1" bit (max power of 2)

1, 2, 4, 8, 16, 32, 64, 128, 256, 512

Subtract powers if possible...

333 – 256 = 77	1
128: no, too big	10
64: yes, subtract: 77 – 64 = 13	101
32: no, too big	1010
16: no, too big	10100
8: yes, subtract: 13 – 8 = 5	101001
4: yes, subtract: 5 – 4 = 1	1010011
2: no, too big	10100110
1: yes, subtract: 1 – 1 = 0	101001101

Binary-to-Decimal Conversion

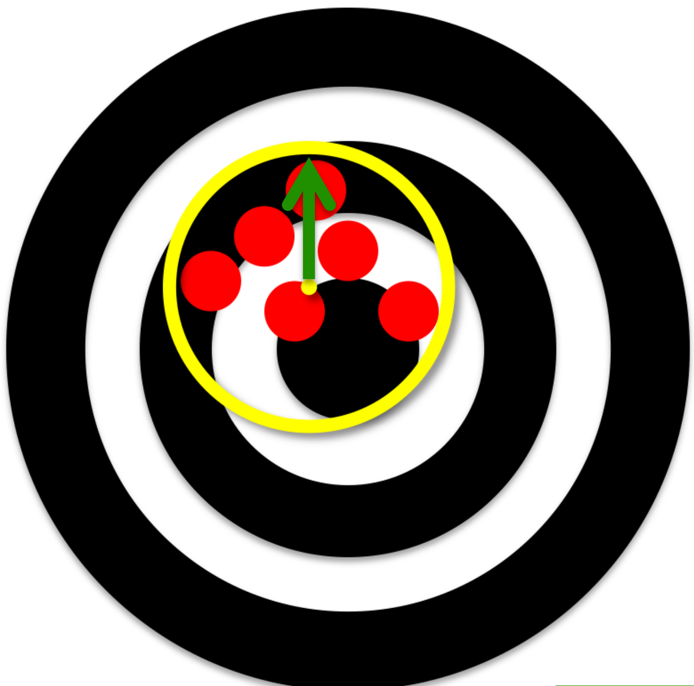
- Count digits (from the right) to the leftmost "1"
  - Note well: the rightmost digit is the "0th"
- Write down the corresponding power of 2
  - Leftmost 1 is k^{th} from right => write down 2^k
  - Example: leftmost 1 is digit 4 => write 16
- Now move right
  - for each 1, write down the corresponding power of 2
- Add up the powers of 2 you wrote down
- Example:

  $\underline{1}00\underline{1}0\underline{1}0 = 2^6 + 2^3 + 2^1 = 64 + 8 + 2 = 74$

4. Interlude: Precision and Accuracy

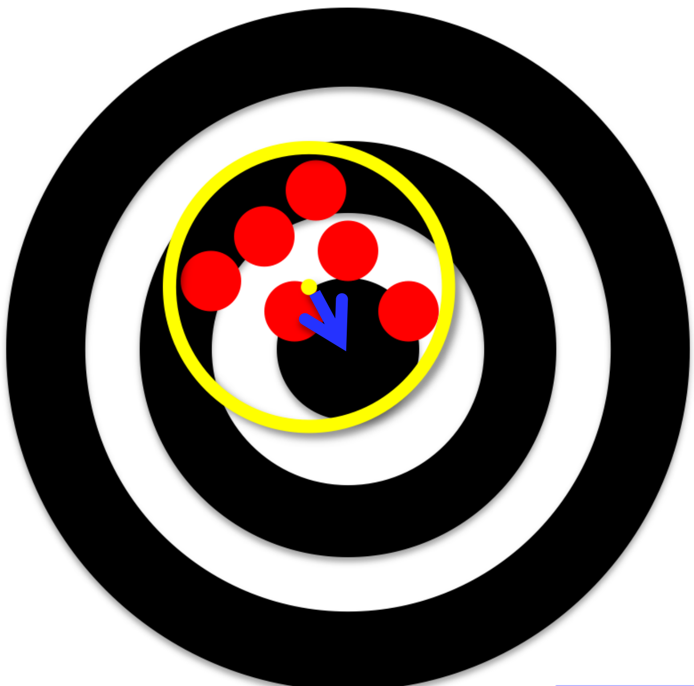
- Precision and accuracy are not the same thing.
  - Both are important in engineering!
- We may consider the precision and accuracy of answers (numbers) obtained various ways:
  - By measurements
  - By calculations
    - Example: predicted maximum height of a projectile
- Precision: how exact is the answer?
- Accuracy: how close to the actual value is the answer?

Precision vs. Accuracy



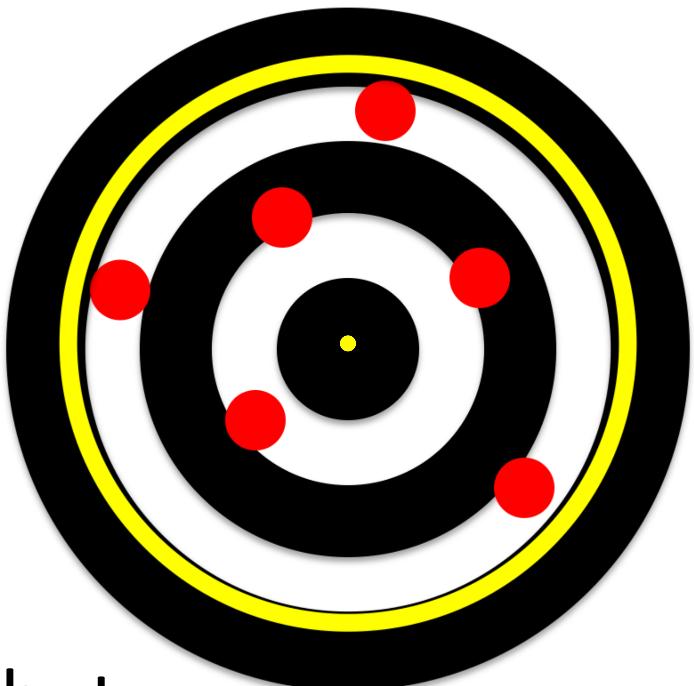
Precision: How tightly shots are grouped (radius of the enclosing circle)

Precision vs. Accuracy



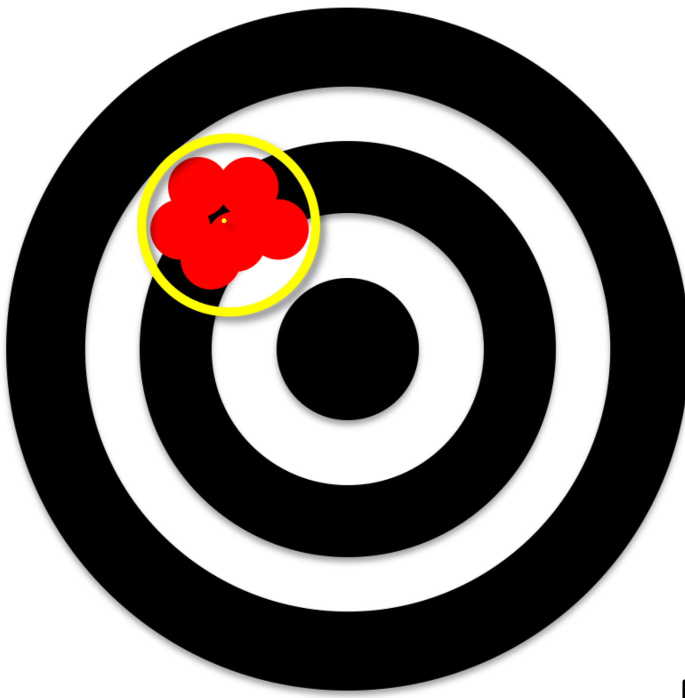
Accuracy: distance from target (center of enclosing circle to bulls-eye)

Precision vs. Accuracy



Accurate, but
not Precise

Precision vs. Accuracy



Precise, but
not Accurate

Significant Digits

- When calculating, significant digits ensure that the precision of a calculation does not exceed that of its inputs
- Significant digits give an idea of the resolution of the measurement/result/prediction
  - 3 significant digits: 10^3 possible values
    - measuring/predicting to 1 part in 1000 (or 0.1%)
  - 10 significant digits: 10^{10} possible values
    - measuring/predicting to 1 part in 10 billion

Significant Digits

- For quantities with decimal points:
  - Significant digits = the number of digits between the leftmost non-zero digit and the rightmost digit (zero or not)
- Examples:

734.999	6 significant digits
734.99913	8 significant digits
0.00012	2 significant digits
0.00012000	5 significant digits
10.00012	7 significant digits

How Many Significant Digits?

- 8345.0010
- -21.77
- 0.000103
- 3.44×10^8
- 200
- 200.0

Rules for Calculations

Note: Excel and MATLAB will not follow these rules!

- Addition and Subtraction: the number of digits to the right of the decimal point should equal the number in the input value with the fewest digits to the right of the decimal point.

  Examples: 6.778 + 3.5 = 10.3

  10.0 – 0.0012 = 10.0

- Multiplication and Division: the number of significant digits in the answer should equal the smallest number of significant digits among the input values.

  Examples: 7.553 X 5.52 = 41.7

  1.0 / 4.5567 = 0.22

Factors Affecting Precision/Accuracy

- Input parameters
  - The output of a calculation cannot be more precise than its inputs! (significant digits)
  - Note that extra digits may be carried along during calculations; rounding occurs at the end
- Modeling assumptions
  - Example: neglecting air resistance in projectile model affects accuracy of the results
- Both: resolution and accuracy of measurements
  - Example: tape measure with 1/16-inch markings

    How many significant digits?
  - Example: systematic error

Bottom Line on Precision

- Report results using precision commensurate with:
  - the precision of all input values
  - the accuracy of any model and/or measurement used
  - Example: It would be wrong to present results of projectile problem with 8 or 9 significant digits (Why?)
- In real engineering practice, 4-5 significant digits will usually suffice

(End of Interlude on Precision and Accuracy)

What About Non-numeric Values?

- Characters are also represented using bytes
  - Need to define a mapping characters <-> bytes
  - Standard ways to do this (why?)
    - ASCII: maps 128 characters (including some non-printable) to 7-bit integers – in the range 0 to 127
    - ISO-8859-1 (a.k.a. "ISO Latin 1") maps characters to 1-byte integers; covers most Western languages; extends ASCII
    - Unicode: maps the symbols in all the world's languages to 16-bit (or more) integers
- Boolean values: true and false
  - In Excel and MATLAB (and many languages) represented by 0=false, nonzero=true

5. Variables, Types, Constants, etc.

- General concept: Computation with software tools is about manipulating values represented as strings of bytes
- These values are stored in the memory of the computer (or on "disk")
- We need to be able to refer to these values so we can tell the computer what to do with them

Referring to Values

- Excel: values are stored in cells
  - Cells are arranged in a large array called a worksheet
    - Multiple worksheets make up a workbook
  - We refer to cells by Column (letter) and Row (number)
    - E.g. B2, D9, AA256, etc.
- MATLAB: each value is stored (independently) in the workspace
  - We refer to values by name
    - E.g., ans, t_new, h, h_new, etc.
  - A named value is called a variable
    - See rules for naming variables in MATLAB help
  - The Workspace Window shows all variables

Types of Variables

- The type of a value tells how the computer interprets the bits in which the value is stored
- The same 4 bytes could be interpreted in different ways
  - Example: 00001010 01111110 00101001 1111111

    or in hexadecimal (base 16): 0a 7e 29 ff

    - signed integer: 176040447
    - single-precision floating-point: 1.22376 e -32
    - Characters: <LF> ~) [0xff is not a character]

Types of Variables, cont.

- In Excel, numerical values are stored as double-precision floating-point numbers
  - You can control the way the value in a cell is displayed: Format -> cells...
  - Non-numeric types (e.g. arbitrary text) are stored as strings of bytes
- In MATLAB, numeric values are double-precision floating-point numbers, by default

Expressions: Computing with Values

As we have seen, the tools can evaluate expressions to compute new values:

- Examples: 10.0 + 0.0012

  t + 0.1

  A8+0.1

h_init + v*t_new*sind(theta)-0.5*g*t_new^2

B2*C6*SIN(B3*PI()/180)-0.5*B1*C6^2

Expressions are built from:

variables (h_init, C6, B2)

constants (180, 0.5, 1.032e7)

operators (+, *, SIN(), ^)

Parsing Expressions

- MATLAB and Excel both interpret, or parse, expressions according to a well-defined set of rules
- Operators are applied in a definite order (precedence) in evaluating an expression:
  - Parenthesized subexpressions are evaluated first
  - then Exponentiation (^)
  - then Unary operators: plus (+), minus (-), logical negation
  - then Multiplication, Division (* , /)
  - then Addition, Subtraction (+ , –)
- Operators with the same precedence are evaluated left-to-right within the expression

Changing Variables' Values

- The power of computing comes from being able to change variables' values in a specified way
  - We say that we assign a value to a variable
  - The old value of the variable is overwritten
- In MATLAB:
  - Assignment statement: <var name> = <expression>
    - Examples: t_new = t + 0.1
      h_new = h_init + v * t_new * sind(theta) - 0.5*g*t_new^2
  - Semicolon after the statement suppresses echoing
  - Variable retains the new value until it is changed by another assignment statement

Changing Variables' Values

- In Excel:
  - A cell containing a formula of the form:

    "=<expression>"

    will always contain the value of <expression>
  - In general, <expression> will refer to other cells
  - Examples:

    =A8 + 0.1

    =B2*C6*SIN(B3*PI()/180)-0.5*B1*C6^2
  - Cell references may be relative, absolute, or mixed

    relative A8 $A8 A$8 A8 absolute

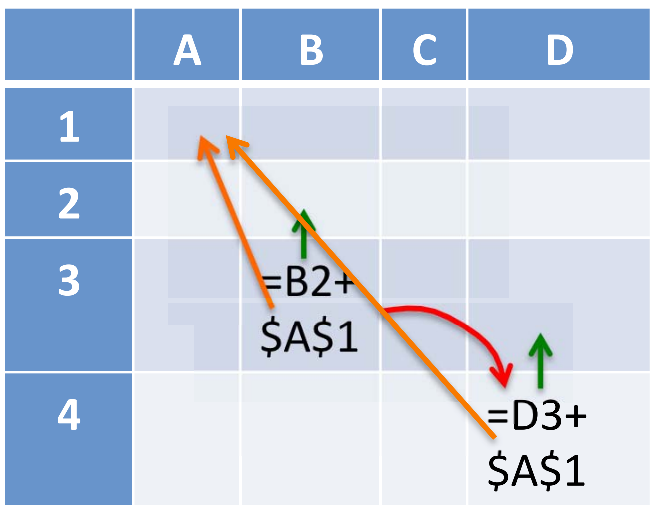
Excel References

- Absolute references don’t change as an expression is copied/moved in a spreadsheet
- Relative references are updated when moved, so they refer to the cell in the same position relative to the new location

	A	B	C	D
1				
2				
3		=B2+ A1		
4				

Excel References

- Absolute references don't change as an expression is copied/moved in a spreadsheet
- Relative references are updated when moved, so they refer to the cell in the same position relative to the new location



Excel and MATLAB Paradigms

- Excel and MATLAB have different computational paradigms
- In MATLAB, computations (programs) are basically sequences of assignment statements
  - Computation proceeds step-by-step
    - Typing assignment statements in the command window
  - Intermediate results are produced, may be output
- In Excel, there is no sequencing
  - Variables (cells) always maintain the same relationship to each other
  - Cells' values change when they depend on other cells whose values change

Excel and MATLAB Paradigms

- Some problems fit more naturally into one paradigm or the other

7. Recap

- Everything inside the computer is represented using 1's and 0's (bits)
- Numbers are represented using binary number system
  - How to convert (whole numbers) back and forth
- There are limits to the values that can be represented
  - Use the "help" function in each tool to find them
- Precision (resolution) vs. Accuracy (correctness)
  - Know factors that affect them: model assumptions, measurement resoution

Recap, (cont.)

- Be aware of significant digits to avoid presenting a result with unwarranted precision
  - Projectile example
  - Software tools will gladly supply 15-digit precision
  - Usually 4 or 5 digits is enough
- Basic programming concepts: variables, expressions, operator precedence
  - MATLAB, Excel precedence rules are similar
  - You should be able to predict the value of any expression you type in MATLAB or Excel
  - When in doubt, parenthesize!

Recap (cont.)

- Assigning a new value to a variable (esp. based on other variables' values) is the basic building block of all computation
- MATLAB and Excel use this building block in different ways
  - MATLAB: sequence assignments in <u>time</u>
  - Excel: <u>maintain relationships</u> among variables (cells) at all times

Lab 2 Preview

- Section 2.2: Entering and Formatting Data
- Section 2.3: Entering and Formatting Formulas
- Section 2.4: Using Built-in Functions