Errors & Source of Errors

Errors in Computing

- Several causes for malfunction in computer systems.
  - Hardware fails
  - Critical data entered incorrectly
  - Software errors
    - Bugs
    - Roundoff
    - Truncation

- Particular to numerical computation
- unavoidable

Numbers on Computers

- The way in which the numbers are presented in the computer is a source of an error.

```
>> 1 - 5*0.2
ans =
    0
>> 1 - 0.2 - 0.2 - 0.2 - 0.2 - 0.2
ans =
    5.5511e-017
```

- What is going on here ?

Numbers on Computers

- Computers use a fixed number of digits to represent a number

- Numerical values stored in computer has finite precision

  ✓ Increasing the speed of numerical calculations

  ✓ Reducing memory required to store numbers

  ✗ Introducing roundoff error

Should be looked how numbers are stored in computers

Bits, Bytes, and Words

- Modern computers manipulate binary numbers (Base 2)
  - Bit: Binary digit – 1 or 0
  - Byte: Group of eight bits

base 10	conversion	base 2
1	$1 = 2^0$	0000 0001
2	$2 = 2^1$	0000 0010
4	$4 = 2^2$	0000 0100
8	$8 = 2^3$	0000 1000
9	$8 + 1 = 2^3 + 2^0$	0000 1001
10	$8 + 2 = 2^3 + 2^1$	0000 1010
27	$16 + 8 + 2 + 1 = 2^4 + 2^3 + 2^1 + 2^0$	0001 1011

Type of Numbers

- Three basic types of numbers:
  - Integers 14
  - Real numbers π
  - Complex numbers $2 + 3i$
- For calculations on computer
  - Numerical values represented by symbols must be stored in computer memory
- Requires translation
  - *symbolic* format (manipulate with pencil & paper)
  - *numeric* format (represented by sequences of bits)

Type of Numbers

- Translation is constrained by number of bytes available to store each type of number.

- Corresponding to

Integers	Integers	limited range of values
Real	Floating-point numbers	limited range & number of decimal digits
Complex	Pairs of floating-point numbers	"

Digital Storage of Integers

MATLAB does not use integer variables

Prelude for discussing floating-point numbers

- Integers can be exactly represented by base 2
- Typical size is 16 bits (2 bytes)
- $2^{16} = 65536$ is largest 16 bit integer
- [-32768, 32767] is range of 16 bit integers
- 32 bit and larger integers are available

Note: All standard mathematical calculations in Matlab use floating point numbers.

Digital Storage of Floating-Point Numbers

Numeric values with non-zero fractional parts are stored as "floating point numbers".

Floating point numbers are stored in a binary equivalent of scientific notation.

All floating point values are represented with a "normalized scientific notation".

Example:

$$12.3792 = \underbrace{0.123792}_{\text{Mantissa}} \times 10^{2}$$

Exponent

Digital Storage of Floating-Point Numbers

- Floating-point numbers are stored as
  - 32-bit values for single precision
  - 64-bit values for double precision
- Floating point values have a fixed number of bits allocated for storage of the mantissa and a fixed number of bits allocated for storage of the exponent.

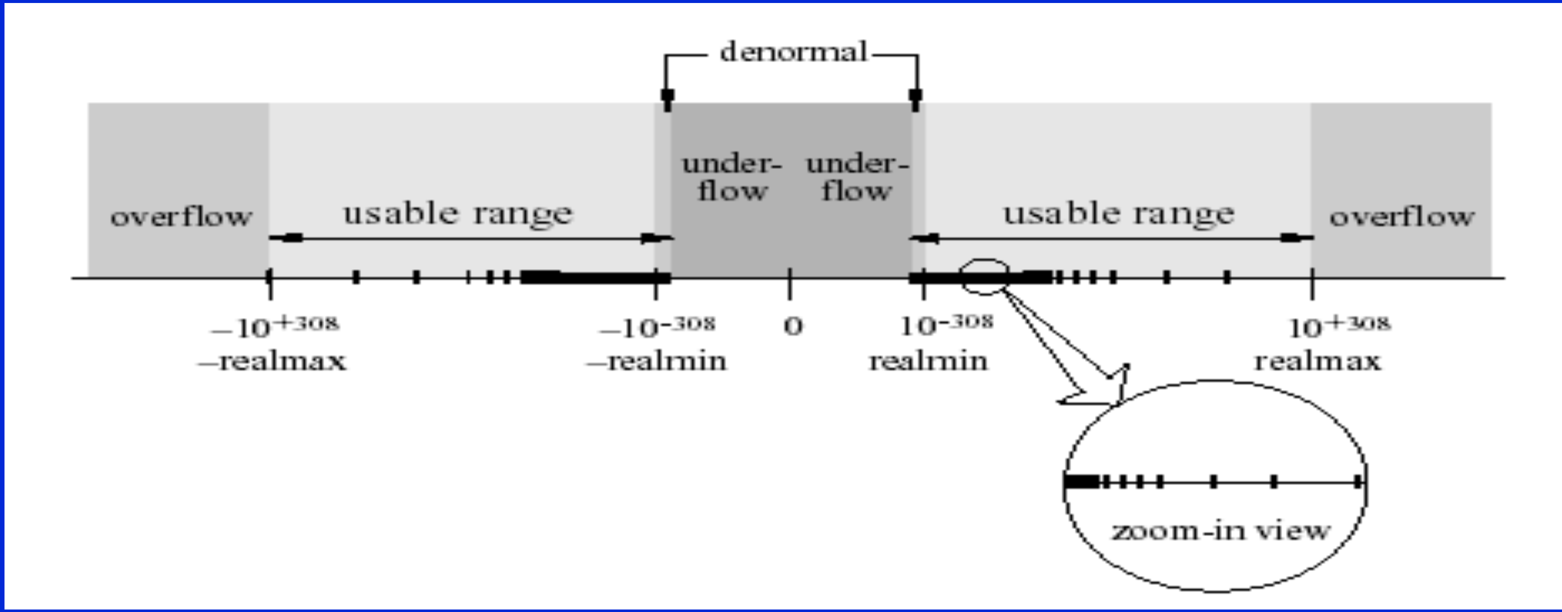
	Bits for		
Precision	Sign	Mantissa	Exponent
Single	1	23	8
Double	1	52	11

Consequences

- Limiting the number of bits allocated for storage of the exponent
  → upper and lower limits on the “range” (magnitude) of floating point numbers
- Limiting the number of bits allocated for storage of the mantissa
  → limit on the “precision” (number of significant digits) for any floating point number.

Floating-Point Number Line

- MATLAB allows numeric variables to be created as 8-bit integers or double-precision floating-point values.
- Most real numbers cannot be stored exactly (they do not exist on the floating-point number line)
- Limitations of 64-bit floating-point values:



Floating-Point Number Line

Compare floating point numbers to real numbers.

- Range
  - Real numbers: Infinite; arbitrarily large and arbitrarily small real numbers exist.
  - Floating point numbers: Finite; the number of bits allocated to the exponent limit the magnitude of floating point values.
- Precision
  - Real numbers: Infinite; there is an infinite set of real numbers between any two real numbers.
  - Floating point numbers: Finite; there is a finite number (perhaps zero) of floating point values between any two floating point values.

!!! Floating-point number line is a subset of the real number line !!!

Roundoff Errors in Computing

- Computers retain a fixed number of significant figures
  - e, π, √7 cannot expressed by fixed number of sig. figures
- Base 2 representations
  - cannot precisely represent exact base 10 numbers
- Discrepancy introduced by omission of significant figures called "roundoff error"
- Effects of roundoff accumulate slowly
- Roundoff errors are inevitable,
  - solution is to create better algorithms
- Subtracting nearly equal may lead to severe loss of precision

Truncation Errors

- Results from approximating continuous mathematical expressions with discrete, algebraic formulas.
- Consider the series for sin(x)

$$\sin(x) = x - x3/3! + x5/5! - x7/7! \cdots$$

- The sine function is defined as infinite series
- For small x, only a few terms are needed to get good approximation to sin(x).
- The $\cdots$ terms are "truncated".

$$f_{true} = f_{sum} + \text{truncation error}$$

- Different than roundoff error, it is under control of user.
- Truncation error can be reduced by selecting more accurate discrete approximations.

Absolute and Relative Errors

- Floating point comparisons should test for “close enough” instead of exact “equality”
  - Don’t ask “is x equal to y”
  - Instead ask, “are x and y ‘close enough’ in value
- “Close enough” can be measured with either absolute difference or relative difference
  - Absolute error $E_{abs} = \hat{e} - e$
  - Relative error $E_{rel} = (\hat{e} - e) / e = E_{abs} / e$

    where

    e = some exact or reference value

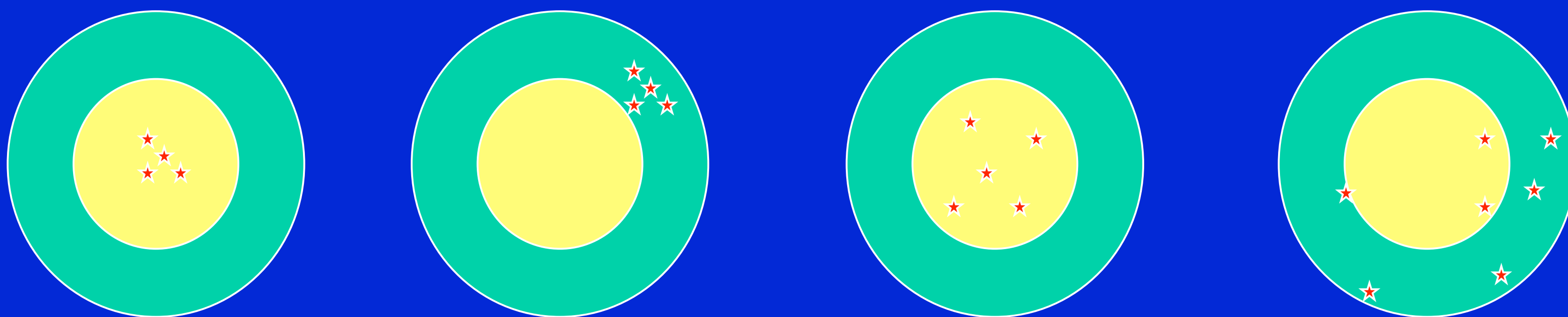
    $\hat{e}$ = some computed value

Precision & Accuracy

- Precision

  The smallest difference that can be represented on the computer (help eps)
- Accuracy

  How close your answer to the “actual” or “real” answer



Good Accuracy
Good Precision

Good Precision
Poor Accuracy

Good Accuracy
Poor Precision

Poor Accuracy
Poor Precision

Low precision: $\pi = 3.14$
High precision: $\pi = 3.140101011$
Low accuracy: $\pi = 3.10212$
High accuracy: $\pi = 3.14159$
High accuracy & precision: $\pi = 3.141592653$