Data Representations

- Computers store data as different variable types, e.g. integer, floating point, complex, etc.
- Different machines have different wordlengths, e.g. 4-byte ints on a 32-bit machine (Pentium), 8-byte ints on a 64-bit machine (Alpha).
- This makes (binary) data non-portable.
Integers

- All data types represented by 0's and 1's.
- An integer value:

\[ j = \sum_{i=1}^{N} s_i \times 2^{N-i} \]

- \( N \) = # of bits in word
- \( s_i \) = value of bit \( i \) in binary string \( s \)

- e.g. 0 0 0 0 0 1 1 0 = \( 2^2 + 2^1 = 6 \) for 8-bit word.
- Use "two's complement" method for sign.
Integers, Cont'd

- Largest value that can be represented is $2^N - 1$.
- For 32-bit word this is 4,294,967,295.
- Arithmetic with integers is exact, except:
  - When division results in remainder.
  - Result exceeds largest representable integer
    e.g. $2 \times 10^9 + 3 \times 10^9 = \text{overflow error}$
- Note multiplication by 2's can be achieved by left-shift, which is very fast (in C: "<<" operator).
Two's Complement

- Exploits finite size of data representations (cyclic groups) and properties of binary arithmetic.
- To get negative of binary number, invert all bits and add 1 to the result.

  e.g. 1 = 0 0 0 0 0 0 0 1 in 8-bit

  invert bits: 1 1 1 1 1 1 1 0
  add 1: 0 0 0 0 0 0 0 1
  result: 1 1 1 1 1 1 1 1 = -1

- In 8 bits, signed char ranges from -128 to +127.
**Negative Powers of 2**

- Binary notation can be extended to cover negative powers of 2, e.g. "110.101" is:
  
  \[ 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^{-1} + 1 \times 2^{-3} = 6.625 \]

- Can represent real numbers by specifying some location in the word as the "binary point" → fixed-point representation.

- In practice, use some bits for an exponent → floating-point representation.
**Floats**

- For most machines these days, real numbers are represented by floating-point format:

\[ x = s \times M \times B^{e-E} \]

- \( s \) = sign \hspace{1cm} B = base (usually 2, sometimes 16)
- \( M \) = mantissa \hspace{1cm} e = exponent
- \( E \) = bias, usually 127.

- In past, manufacturers used different number of bits for each of \( M \) and \( e \) → non-portable code.
Floats, Cont'd

- Currently, most manufacturers adopt IEEE standard:
  - $s = 1^{st}$ bit
  - Next 8 bits are $e$
  - Last 23 bits are $M$, expressed as a binary fraction, either $1.F$, or, if $e=0$, $0.F$, where $F$ is in base 2.

- Largest single-precision float $f_{\text{max}} = 2^{127} \approx 10^{38}$.

- Smallest (and least precise!) $f_{\text{min}} = 2^{-149} \approx 10^{-45}$. 
Round-off Error

- Not all values along real axis can be represented.
- There are $10^{38}$ integers between $f_{\text{min}}$ & $f_{\text{max}}$, but only $2^{32} \approx 10^9$ bit patterns.
- Values $< |10^{-45}|$ result in "underflow" error.
- If value cannot be represented, next nearest value is produced. Difference between desired and actual value is called "round-off error" (RE).
Round-off Error, Cont'd

- Smallest value $e_m$ for which $1 + e_m > 1$ is called "machine accuracy", typically $\sim 10^{-7}$ for 32 bits.

- Double precision greatly reduces $e_m$ ($\sim 10^{-16}$).

- RE accumulates in a calculation:
  - Random walk: total error $N^{1/2} e_m$ after $N$ operations.
  - But algorithms rarely random $\rightarrow$ linear error $N e_m$.
Round-off Error, Cont'd

- Subtraction of two very nearly equal numbers can give rise to large RE.
  
  e.g. Solution of quadratic equation...

  \[ x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \]

  ...can go badly wrong whenever \( ac \ll b^2 \) (Cf. PS#2).

- RE cannot be avoided—it is a consequence of using a finite number of bits to represent real values.
Truncation Error

- In practice, most numerical algorithms approximate desired solution with a finite number of arithmetic operations.
  - e.g. evaluating integral by quadrature
  - summing series using finite number of terms
- Difference between true solution and numerical approximation to solution is called "truncation error" (TE).
Truncation Error, Cont'd

- TE exists even on "perfect" machine with no RE.
- TE is under programmer's control; much effort goes into reducing it.
- Usually RE and TE do not interact.
- Sometimes TE can amplify RE until it swamps calculation. Solution is then called unstable.

  e.g. Integer powers of Golden Mean (Cf. PS#2).