Scientific Computing

Is an important part of of Computational Science that covers:

- Development of mathematical models (equations),
- **Development of algorithms to solve equations numerically,**
- **Implementation of algorithms in software,**
- **Numerical simulation of physical phenomena using computer software,**
- Interpretation, Validation and Visualisation of results.
- Redesign needed?

History of SC

- Most of the concepts formulated 200 years ago by Newton, Gauss, Euler, Jacobi and many others.
- The motivation was obtaining approximate solutions for mathematical problems that arose in physics, astronomy and other fields of science.
- Efficient use of computational resources (pencil, paper, brain power),
- with the advent of computers the problem sizes are increasing,
- rounding errors are becoming critical, because the precision is not under human control,
- computation (simulation of reality) is becoming as important as measurements and theory.

Propagation Error

- Let e express the relative error in representing a nonzero floating point number.
- The sum: $a(1 \pm e) + b(1 \pm e) = (a \pm ae) + (b \pm be) =$
  $= (a + b) \pm e\,(a + b) = (a+b)\,(1 \pm e)$

  The sum error is in the same range as the error of factors. Similar is valid for difference, but suppose that a and b are of similar values then a-b $=^{app}$ 0 and e may become as large as result!
- The product: $a(1\pm e)\,.\,b(1 \pm e) = (a \pm ae)\,.\,(b \pm be) =$
  $= ab \pm abe \pm bae \pm abe^2 =^{app} ab(1 \pm 2e)$

  We get double error what leads to the pessimistic estimation of propagation error. Similar is valid for division $((1 \pm e)^{-1} = 1 \pm e + e^2 \pm \ldots =^{app} 1 \pm e)$.

Propagation Error (Cont.)

- For an exponent of n the relative error of result is n-times greater than initial error:
  $a(1 \pm e)^n = a^n\,[(1 \pm ne \pm n(n-1)e^2/2! \pm \ldots] =^{app} a^n\,(1 \pm ne)$
- What happens if the exponent is smaller than 1? Is the error in result smaller? No.
- Similar is valid for m consecutive multiplication or division operations. Error can increase in the worst case for a factor of m.
- But in practical consecutive calculations we usually desire that the final result should have the similar absolute error as the less accurate input data.

Floating-Point Numbers

- Floating-point number system represents approximately the real number system.
- Floating point numbers are used in a similar way as scientific notation, with an exponent.
- Examples:
  $2347 = 2.347 * 10^3$,
  $0.0007396 = 7.396 * 10^{-4}$.
- The name floating-point is used because the decimal point floats as the power of 10 changes.

Floating-Point Numbers IEEE SP-normalized

- radix or base $b = 2$,
- precision p=24 (23 bits for mantissa, 1bit for sign)
- exponent range [L=-126, U=127] (8 bits for exponent)
- 32 bits used for representation,
  $(1\ \ 00110001101001001001101\ \ 10100101)_2 =$
  $= -\,(2^{-3} + 2^{-4} + 2^{-8} + 2^{-9} + 2^{-11} + 2^{-14} + 2^{-17} + 2^{-20} + 2^{-21} + 2^{-23}) * 2^{-37} =$
  $(-\,1.41113822957573241012596554355696 * 10^{-12})_{10}$
- the least significant bit in mantissa 1->0, the gap between two consecutive numbers:
  $2^{-23} * 2^{-37} = 2^{-60} =$
  $= 8.67361737988403547205962240695 * 10^{-19}$
- gaps are equally spaced between powers of b, but become smaller and smaller if we approaching to zero.

Floating-Point Numbers IEEE SP-normalized (Cont.)

- The number of all floating point numbers:
  $2 * (b-1) * b^{p-1} * (U-L+1) + 1 = 2^{24} * 254 + 1 =$
  $4.261412865 * 10^{9}$
- Underflow (the smallest positive normalized number):
  $1.0 * b^{L} = 1.0 * 2^{-126} =$
  $= 1.17549435082228750796873653722 * 10^{-38}$
- Overflow level (the largest possible number):
  $b^{U+1} * (1 - b^{-p}) = 1.1111... * 2^{127} =$
  $= 3.40282346638528859811704183484 * 10^{+38}$

Calculation of the Overflow level

- sum of geometric series:
  $d_0 + d_0 * q + d_0 * q^2 + ... + d_0 * q^n$
  is: $Sn = d_0 * (1 - q^n) / (1 - q)$

- floating point representation of the largest number:
  $(d_o + d_1 b^{-1} + d_2 b^{-2} + ... + d_{p-1} b^{-(p-1)}) * b^U =$
  $= ((b-1) + (b-1)b^{-1} + (b-1)b^{-2} + ... + + (b-1)b^{-(p-1)}) * b^U$
  we have: n=p, $d_0 = (b-1)$, and $q = b^{-1}$ so the sum is:

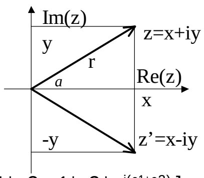
  $((b-1) * (1 - b^{-p}) / (1 - b^{-1})) * b^U =$
  $= ((b-1) * b * (b^p - 1) / (b-1) * b^p) * b^U =$
  $= b^{-(p+1)} * (b^p - 1) * b^U = b^{U+1} (1 - b^{-p})$

Machine Precision

- Determines the maximum possible relative error in representing a nonzero real number.
- We know that the smallest positive normalized number is:
  $1.0 * b^L = 1.0 * 2^{-126} =$
  $= 1.17549435082228750796873653722 * 10^{-38}$
- The nearest neighbouring number:
  $1.00000000000000000000000001 * b^L = (1.0 + 2^{-23}) * 2^{-126} =$
  $= 1.17549449095213394045044362 95 * 10^{-38}$
- The difference and relative error (constant for all FP numbers):
  $diff = 1.401 * 10^{-45}$, rel_err = $e_{mach} = 1.192 * 10^{-7}$
  what means that we have about 7 digits of precision in SP.
- Machine precision with rounding to nearest in IEEE SP: $1 * b^{1-p} = 1 * 2^{-23} = 5.9604644775390625 * 10^{-8}$

Complex Numbers

- A given complex number z can be expressed as
  z=x + i*y, where x=Re(z) and y=Im(z) are real numbers and i=sqrt(-1).
- x and y are real and imaginary parts, respectively, of z. A complex number can be real if Im(z)=0 or purely imaginary if Re(z)=0.
- Complex conjugate $\bar{z}$=x-iy
  $z * \bar{z} = x^2 + y^2$ r=|z|=sqrt(x^2+y^2)
- Complex arithmetic more costly than real arithmetic.
  z1*z2=(x1x2-y1y2)+i(x1y2+x2y1)]
- Polar presentation:
  z=r*(cosa + i*sina) = r*e^{ia} [now: z1*z2=r1*r2*$e^{i(a1+a2)}$]



Some remarks on Software

- FORTRAN (reliable, tuned, efficient BUT expensive, restrictive, not available for all machines),
- C, C++ (free, versatile , popular, available on all platforms) ,
- TRANSLATORS (f2c) opposite array storage convention, function calls,
- MATLAB (high-level, interactive, built-in graphic, easy to learn)
- Basic information on MatLab in Getstart.pdf
- Run MatLab with: `/usr/common/matlab-13/bin/matlab`
- First practicing during Labs.

Different Languages in Ongoing (19079) Open Source Projects (Sept.03) http://freshmeat.net

C	28.55 %
Perl	14.42 %
C++	12.80 %
Java	12.22 %
PHP	10.59 %
Python	6.14 %
Unix Shell	2.88 %
Tcl	1.87 %
SQL	1.54 %
JavaScript	1.24 %
Fortran	0.24 %
Ada	0.20 %
Basic	0.08 %

...