Representing Numbers on the Computer

Let's calculate J!

```
 Program Maxnumber
*
*   Check what the max numbers are on the computer
*
      Integer N

*
      N=1
      Do J=1,20
         N=N*J
         Print *,J,N
      Enddo
*
      stop
      end
```

What happened?

J	N=J!
1	1
2	2
3	6
4	24
5	120
6	720
7	5040
8	40320
9	362880
10	3628800
11	39916800
12	479001600
13	1932053504
14	1278945280
15	2004310016
16	2004189184
17	-288522240
18	-898433024
19	109641728
20	-2102132736

Representing Integers

E.g., single precision: 4 bytes or 32 bits

1 bit is used for the sign (1 for - 0 for +)
31 bits for value

Because start from 0

Biggest integer $2^{31}-1$ = 2 147 483 647= 01111111 11111111 11111111 11111111

2's complement is standard for integer representation.

8 bit example (from Wikipedia)

Sign Bit								Value
0	1	1	1	1	1	1	1	127
0	0	0	0	0	0	1	0	2
0	0	0	0	0	0	0	1	1
0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	-1
1	1	1	1	1	1	1	0	-2
1	0	0	0	0	0	0	1	-127
1	0	0	0	0	0	0	0	-128

Representing Integers

To calculate the 2's complement value: $N^{*}=2^{n}-N$, where n is the number of bits used to represent an integer. N^{*} is the 2's complement representation of the negative of N.

e.g., $N_{10}=5$ $N_{2}=0000\ 0101$ (n=8)
$N^{*}_{2}=2^{n}-N_{2}=1\ 0000\ 0000 - 0000\ 0101 = 1111\ 1011$

2's complement is convenient for computer calculations

There is no rounding error - only a maximum allowed range for the values.

For the mathematicians: 2^{n} possible values of n bits form a ring of equivalence classes

Representing Integers

Or, invert the bits and add 1.

E.g., 5 = 0000 0101

To convert to -5, flip the bits ⇒ 1111 1010

Then add 1 ⇒ 1111 1011

The other way, to go from -5 to 5,
flip the bits ⇒ 0000 0100
And add 1 ⇒ 0000 0101

Representing Real Numbers

Representation of real numbers (scientific notation - IEEE754):

Mantissa and Exponent + sign bit. E.g., single precision (4 bytes)

1	+	8	+	23	= 32 bits
(sign)		(exponent)		(mantissa)	

Double precision

1	+	11	+	52	= 64 bits
(sign)		(exponent)		(mantissa)	

$$x = (-)^{s} \bullet a \bullet 2^{b-E}$$

s is sign bit

a is normalized so first bit is 1 (radix point - implicit)

E = 1/2 of (maximum exponent -1), or

E=01111111 in single precision

Example: 4/7 on the computer:

$$\left.\frac{4}{7}\right|_{10} = 1.001001001\cdots2^{-1}\Big|_2$$

$$\left.\frac{4}{7}\right|_{10} = \left.\frac{100}{111}\right|_2 = 0 + \frac{1000}{111}2^{-1} = 0 + 1\cdot2^{-1} + \frac{1}{111}\cdot2^{-1} =$$

$$= 0 + 1\cdot2^{-1} + 0\cdot2^{-2} + 0\cdot2^{-3} + \frac{1000}{111}\cdot2^{-4} = 0.1001001001\cdots = 1.001001001\cdots2^{-1}$$

$$s = 0$$

$$a = 001001001001001001 \quad \text{(first 1 is implicit)}$$

b-E=-1 or, $b_2 - 0111111 = -1|_{10}$, $b_2 = 01111110$

Representing Real Numbers

If the exponent $b=11111111$, the number has a special value:

- if $a=00000000000000000000000$, value is $\pm\infty$ depending on s
- else, value is NaN (not a number)

If $b=00000000$

- $x=\pm 0.a \bullet 2^{-126}$

Otherwise $x=\pm 1.a \bullet 2^{b-127}$ (single precision)

Precision	# bits		Relative precision	Max magnitude	Min magnitude (normalize d)
	a	b			
single	23	8	$2^{-23}\approx 10^{-7}$	$2^{(255-127)}\approx 10^{38}$	$\approx 10^{-38}$
double	52	11	$2^{-52}\approx 10^{-16}$	$2^{(2047-1023)}\approx 10^{308}$	$\approx 10^{-308}$

Calculation of π

As an example, consider the calculation of π using the following algorithm (due to Madhava of Sangamagrama, Indian Mathematician of the 14th century)

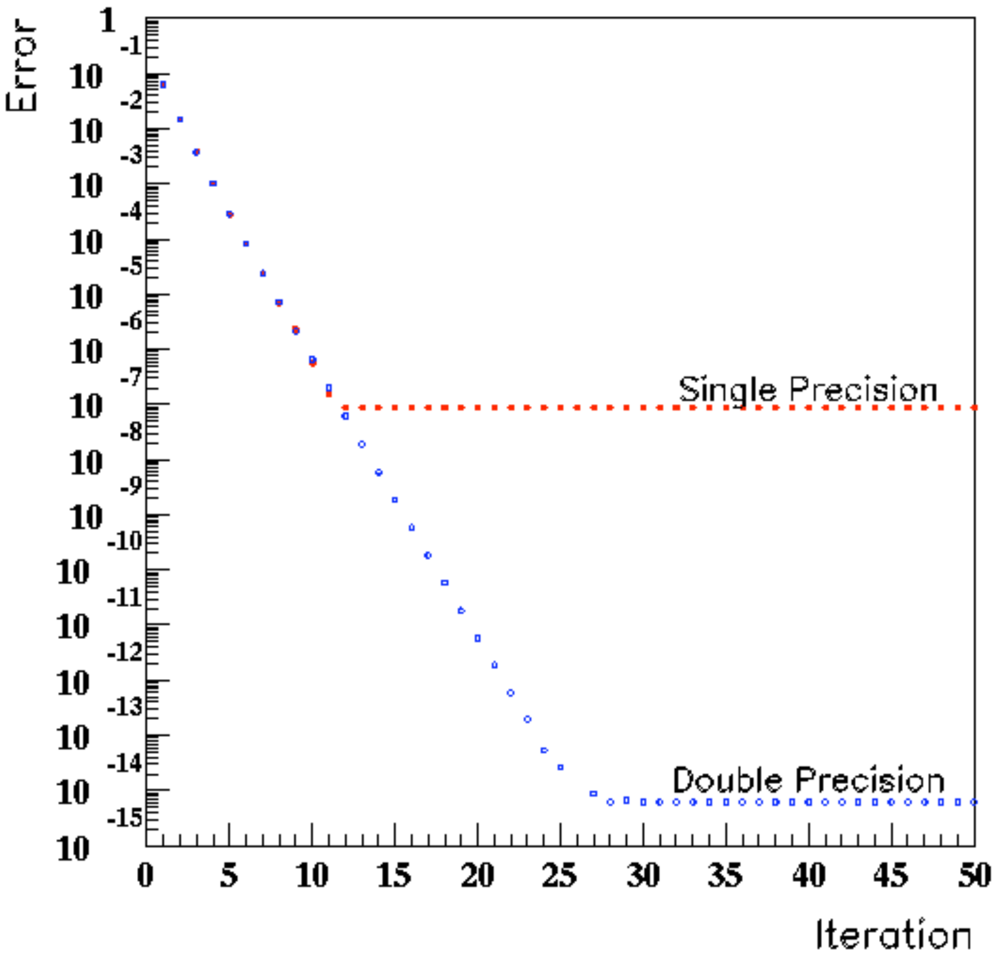
$$\pi = \sqrt{12}\sum_{i=0}^{\infty}(-1)^i \frac{1}{(2i+1)3^i}$$

I	*single precision*	*double precision*
1	3.079201459885	3.079201435678
2	3.156181335449	3.156181471570
3	3.137852907181	3.137852891596
4	3.142604827881	3.142604745663
5	3.141308784485	3.141308785463
6	3.141674280167	3.141674312699
7	3.141568660736	3.141568715942
8	3.141599655151	3.141599773812
9	3.141590356827	3.141590510938
10	3.141593217850	3.141593304503
11	3.141592502594	3.141592454288
12	3.141592741013	3.141592715020
13	3.141592741013	3.141592634547
20	3.141592741013	3.141592653596
Error	⇑	⇑

First 16 digits of correct value
3.14159 26535 89793

After 20 iterations, single precision good to 1 10^{-7} Double precision to 10^{-11}

Calculation of π



Close to 10^{-16}

Calculation of π

Dear folks, 20th October 2005

Our latest record which was announced already at press release time of 6-th of December, 2002 was as the followings;

Declared record: **http://www.super-computing.org/pi_current.html**

1,030,700,000,000 hexadecimal digits
1,241,100,000,000 decimal digits

Two independent hexadecimal calculation based on two different algorithms generated more than 1,030,775,430,000 hexadecimal digits of pi and comparison of two generated sequences matched completely. Computed hexadecimal digits of pi were radix converted into base 10, generating more than 1,241,177,300,000 decimal digits of pi and generated decimal digits of pi were radix converted again into base 16. Radix converted hexadecimal digits of pi were compared with original hexadecimal digits of pi. There were no difference up to 1,241,100,000,000 decimal digits. Then we are declaring 1,030,700,000,000 hexadecimal digits and 1,241,100,000,000 decimal digits as the new world records. Details of computed results are available on the following URL's.

http://www.super-computing.org/pi-hexa_current.html (hexadecimal)
http://www.super-computing.org/pi-decimal_current.html (decimal)

Rounding Errors for Simple Sum

In contrast to integers, there are rounding errors for real numbers.The error resulting from adding two numbers:

$$y = x_1 + x_2$$

$$\tilde{y} = rd\left[rd(x_1) + rd(x_2)\right] \text{ where } rd() \text{ means computer rounding}$$

$$\tilde{y} \approx \left[x_1(1+\varepsilon) + x_2(1+\varepsilon)\right](1+\varepsilon) \text{ where } \varepsilon \text{ is the typical relative error}$$

$$|\varepsilon| \approx 2^{-t} \text{ where } t \text{ is the number of bits assigned to the mantissa}$$

$$\text{single precision, } \varepsilon = 2^{-23} \approx 10^{-7} \quad \text{double precision, } \varepsilon = 2^{-52} \approx 10^{-16}$$

$$\tilde{y} \approx x_1 + x_2 + \varepsilon(x_1 + x_2) + x_1\varepsilon_1 + x_2\varepsilon_2$$

$$\frac{\tilde{y} - y}{y} \approx \varepsilon + \frac{x_1}{x_1 + x_2}\varepsilon_1 + \frac{x_2}{x_1 + x_2}\varepsilon_2$$

Can get large multiplication of relative error if $x_1 \approx -x_2$

Error Propagation

More generally (see Lecture Notes from Scherer):

Input data $\vec{x} = (x_1, \cdots, x_n)$

Output data $\vec{y} = (y_1, \cdots, y_m)$

where

$\vec{y} = \varphi(\vec{x}) = \varphi^{(r)}\varphi^{(r-1)} \cdots \varphi^{(1)}$ and the φ are simple functions

Define

$$\vec{x}_1 = \varphi^{(1)}(\vec{x})$$

$$\vec{x}_i = \varphi^{(i)}(\vec{x}_{i-1})$$

$$\vec{y} = \varphi^{(r)}(\vec{x}_{r-1})$$

Treat all errors as small, represent with $\Delta\vec{x}$

Error Propagation

First step:

$$\tilde{\tilde{\vec{x}}}_1 = rd(\varphi^{(1)}(\vec{x} + \Delta\vec{x})) \approx \left(\varphi^{(1)}(\vec{x}) + D\varphi^{(1)}\Delta\vec{x}\right)(1 + E_1)$$

First order in errors

where

$$D\varphi^{(1)} = \left(\frac{\partial x_{1i}}{\partial x_j}\right) = \begin{pmatrix} \frac{\partial x_{11}}{\partial x_1} & \cdots & \frac{\partial x_{11}}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial x_{1n_1}}{\partial x_1} & \cdots & \frac{\partial x_{1n_1}}{\partial x_n} \end{pmatrix}$$

and

$$E_1 = \begin{pmatrix} \varepsilon_1^{(1)} & & \\ & \ddots & \\ & & \varepsilon_{n_1}^{(1)} \end{pmatrix}$$

$$\Delta\vec{x}_1 = \tilde{\tilde{\vec{x}}}_1 - \vec{x}_1 \approx D\varphi^{(1)}\Delta\vec{x} + \varphi^{(1)}(\vec{x})E_1$$

Error Propagation

$$\Delta\vec{y} \approx \vec{y}E_r + D\varphi^{(r)}\cdots\varphi^{(1)}\Delta\vec{x} + D\varphi^{(r)}\cdots\varphi^{(2)}\vec{x}_1E_1 + \cdots + D\varphi^{(r)}\vec{x}_{r-1}E_{r-1}$$

$$D\varphi = D\varphi^{(r)}\cdots\varphi^{(1)} = \begin{pmatrix} \frac{\partial y_1}{\partial x_1} & \cdots & \frac{\partial y_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_m}{\partial x_1} & \cdots & \frac{\partial y_m}{\partial x_n} \end{pmatrix}$$

The first term is the inevitable rounding error
The second term contains the propagation of the input errors and initial rounding errors.
The other terms depend on how the algorithm is set up.

Error Propagation

Let's look at the individual terms:

$$\vec{y}E_r\big|_i \approx |y_i|\varepsilon$$

The rounding error on the final answer

$$D\varphi^{(r)}\cdots\varphi^{(1)}\Delta\vec{x}\big|_i \approx \sum_j \left|\frac{\partial y_i}{\partial x_j}\right| |\Delta x_j|$$

Propagation of input errors

The other terms depend on the specific algorithm. The goal is for the algorithm to not give errors larger than the first two (unavoidable) errors.

Error Propagation

Let us look at an example in detail - the calculation of a^2-b^2

Procedure I:

1. Calculate a^2 and b^2
2. Calculate their difference

$$\vec{x} = \begin{pmatrix} a \\ b \end{pmatrix} \qquad \vec{x}_1 = \begin{pmatrix} x_1^2 \\ x_2^2 \end{pmatrix} \qquad \vec{y} = x_{11} - x_{12}$$

Unavoidable error:

$$|y|\varepsilon = \left|a^2 - b^2\right|\varepsilon \qquad \sum_j \left|\frac{\partial y}{\partial x_j}\right|\left|\Delta x_j\right| = \left|\frac{\partial(a^2-b^2)}{\partial a}\right|\varepsilon + \left|\frac{\partial(a^2-b^2)}{\partial b}\right|\varepsilon = 2\left(|a|+|b|\right)\varepsilon$$

$$\Delta y^{(0)} = \left|a^2 - b^2\right|\varepsilon + 2\left(|a|+|b|\right)\varepsilon$$

Error Propagation

Let us look at an example in detail - the calculation of a^2-b^2

Procedure I:

1. Calculate a^2 and b^2
2. Calculate their difference

$$\vec{x} = \begin{pmatrix} a \\ b \end{pmatrix} \qquad \vec{x}_1 = \begin{pmatrix} x_1^2 \\ x_2^2 \end{pmatrix} \qquad \vec{y} = x_{11} - x_{12}$$

Error magnitude estimation:

$$\tilde{\vec{x}} = \begin{pmatrix} a(1+\varepsilon_a) \\ b(1+\varepsilon_b) \end{pmatrix} \qquad \tilde{\vec{x}}_1 = \begin{pmatrix} a(1+\varepsilon_a)a(1+\varepsilon_a)(1+\varepsilon_{11}) \\ b(1+\varepsilon_b)b(1+\varepsilon_b)(1+\varepsilon_{12}) \end{pmatrix} \approx \begin{pmatrix} a^2(1+2\varepsilon_a+\varepsilon_{11}) \\ b^2(1+2\varepsilon_b+\varepsilon_{12}) \end{pmatrix}$$

$$\tilde{\vec{y}} = \left[a^2(1+2\varepsilon_a+\varepsilon_{11}) - b^2(1+2\varepsilon_b+\varepsilon_{12}) \right](1+\varepsilon_2)$$

$$|\Delta y| \le \left| a^2 - b^2 \right| \varepsilon + 3(a^2+b^2)\varepsilon$$

Error Propagation

Procedure II:

1. Calculate *a-b* and *a+b*
2. Calculate their product

$$\vec{x} = \begin{pmatrix} a \\ b \end{pmatrix} \qquad \vec{x}_1 = \begin{pmatrix} x_1 - x_2 \\ x_1 + x_2 \end{pmatrix} \qquad \vec{y} = x_{11} \bullet x_{12}$$

Error magnitude estimation:

$$\tilde{\vec{x}} = \begin{pmatrix} a(1+\varepsilon_a) \\ b(1+\varepsilon_b) \end{pmatrix} \qquad \tilde{\vec{x}}_1 = \begin{pmatrix} \big(a(1+\varepsilon_a) - b(1+\varepsilon_b)\big)(1+\varepsilon_{11}) \\ \big(a(1+\varepsilon_a) + b(1+\varepsilon_b)\big)(1+\varepsilon_{12}) \end{pmatrix} \approx \begin{pmatrix} (a-b)(1+\varepsilon_{11}) + a\varepsilon_a - b\varepsilon_b \\ (a+b)(1+\varepsilon_{12}) + a\varepsilon_a + b\varepsilon_b \end{pmatrix}$$

$$\tilde{\vec{y}} = \left[(a^2 - b^2)(1+\varepsilon_{11}+\varepsilon_{12}) + 2a^2\varepsilon_a - 2b^2\varepsilon_b \right](1+\varepsilon_2)$$

$$\left|\Delta y\right| \le 3\left|a^2 - b^2\right|\varepsilon + 2(a^2+b^2)\varepsilon$$

Error Propagation

Single precision

a	b	Exact value (a^2-b^2)	a^2-b^2	(a-b)(a+b)
1.0	0.999	1.999 10^{-3}	1.99896 10^{-3}	1.99897 10^{-3}
1.0	0.9999	1.9999 10^{-4}	2.00033 10^{-4}	2.00023 10^{-4}

Exercises 2

1. Look up a different algorithm to calculate π from the one presented in the lecture and code it in single and double precision. Compare the speed of convergence to the one shown in class.

2. Calculate (a^4-b^4) numerically in single and double precision. Compare the resulting accuracy to the true value for test cases. Compare to the expected precision for single and double precision calculations.