Unit 4: Single vs double precision: accuracy of floating-point arithmetic

If a real number, x, is approximated and stored using floating-point format

$$fl(x) = (-1)^s \left(1.0 + \sum_{n=1}^{p-1} a_n 2^{-n}\right) 2^{e-bias},$$

then $fl(x)$ satisfies

$$fl(x) = x(1+\delta),$$

where $|\delta| \leq 2^{-p}$. The values of 2^{-p} for different precisions are given in decimal format (to 8 significant figures) in the table below.

Common name	$\boldsymbol{p}$	$\boldsymbol{2^{-p}}$
Half precision	11	4.8828125×10^{-4}
Single precision	24	5.9604645×10^{-8}
Double precision	53	$1.1102230 \times 10^{-16}$
Quadruple precision	113	$9.6296497 \times 10^{-35}$
Octuple precision	237	$4.5278395 \times 10^{-72}$

As well as recognising the underlying error in the storage of real numbers, it is also important to consider the accuracy of basic arithmetic operations. Processors that satisfy the IEEE standard 754 will perform floating-point operations that satisfy the Standard Model of Arithmetic:

Let x and y be floating-point format numbers, then

$$fl(x \text{ op } y) = (x \text{ op } y)(1+\delta), \quad |\delta| \leq 2^{-p}, \quad \text{op} = +, -, *, /.$$

Example

Suppose that real numbers x and y are stored using floating-point format and then added together. Bound the overall error of the resulting floating point number relative to $x + y$.

Now $fl(x) = x(1+\delta_x)$ and $fl(y) = y(1+\delta_y)$, where $|\delta_x| \leq 2^{-p}$ and $|\delta_y| \leq 2^{-p}$. Hence,

$$\begin{aligned} fl(fl(x) + fl(y)) &= (fl(x) + fl(y))(1+\delta) \\ &= (x(1+\delta_x) + y(1+\delta_y))(1+\delta) \end{aligned}$$

In the worst case, $\delta = \delta_x = \delta_y = 2^{-p}$. This gives

$$\begin{aligned}(x(1+\delta_x)+y(1+\delta_y))(1+\delta) &= (x(1+2^{-p})+y(1+2^{-p}))(1+2^{-p}) \\ &= (x+y)(1+2^{-p})(1+2^{-p}) \\ &= (x+y)(1+2^{-p}+2^{-p}+2^{-2p}) \\ &= (x+y)(1+2^{1-p}+2^{-2p}).\end{aligned}$$

We therefore obtain

$$fl\big(fl(x)+fl(y)\big) = (x+y)(1+\epsilon),$$

Where $|\epsilon| \leq 2^{1-p}+2^{-2p}$.

The above example demonstrates how error can easily start to accumulate. In most simulations, repeated arithmetic operations are made and, hence, there are cases where the use of single precision instead of double precision is not appropriate. It is also important to consider how the size of the number being operated on fit into the floating-point format being used. For example, the addition of a very large number with a very small number.

Example

The following Octave code takes two numbers, s and t, and an integer n and forms $r = s + n \times t$ by using a loop to repeatedly add t.

```
function r=prec_add(s,t,n)

  r=s;
  for i=1:n
    r=r+t;
  endfor

endfunction
```

The calls r=prec_add(single(2^10),single(2^-14),1000) and r=prec_add(2^10,2^-14,1000) return 1024 and 1024.061035156250, respectively. In exact arithmetic

$$2^{10} + 2^{-14} = 2^{10}(1.0 + 2^{-24}).$$

This can be exactly represented in double precision. In single precision, the 2^{-24} is smaller than the smallest value of 2^{-p} that can be stored in the fraction and, hence, in single precision, $2^{10} + 2^{-14}$ is truncated to 2^{10}. No matter how many times the loop is executed, the resulting value of r will always be 1024 for the single precision version. Let the order of the operations be switched such that t is added to r at the end of the function:

```
function r=prec_add1(s,t,n)

r=0;
for i=1:n
r=r+t;
endfor
r=r+s;

endfunction
```

The calls r=prec_add1(single(2^10),single(2^-14),1000) and r=prec_add1(2^10,2^-14,1000) return 1024.0610 and 1024.061035156250, respectively.

Whilst the above example may be considered to be rather contrived. It is not uncommon for application codes to require summations where a large number of (relatively) small values are added to (relatively) large values, e.g., molecular dynamics codes. Whilst, ideally, the values should be summed in an order that will maximise the accuracy of the overall calculation (see Chapter 4, [1]), it is not normally practical to order the summation to ensure that this happens and, hence, loss in accuracy can easily occur if care is not taken to used the correct level of precision. We were able to demonstrate this loss in accuracy in the metal potential calculations within DL_POLY, where double precision is necessary to produce an accurate enough result [2].

[1] "Accuracy and Stability of Numerical Algorithms (Second Edition)", *N. J. Higham*, SIAM, 2002.
[2] "Using mixed precision within DL_POLY's force and energy evaluations: short-range two-body interactions", *H. S. Thorne*, RAL Technical Report RAL-TR-2018-004, 2018.