Unit 7: Conclusions

This course has introduced IEEE floating point numbers (IEEE 754 standard), which take the form

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

We discussed how single precision floating-point arithmetic may be advantageous over double precision with respect to execution time and energy consumption. The advantages with respect to data movement were also considered. However, there are overheads from precision conversions.

Whilst the potential of reduced execution time and energy consumption is tempting, it is important that the loss in accuracy of computations is taken into account. In Unit 4, we bounded the errors made during each floating-point operation and considered some examples that showed how careful one must be to minimise the loss in accuracy. Most algorithms have had some analysis done to determine how the nature of the underlying problem being solved and the precision of the floating-point arithmetic affects the accuracy of the computation. These results can be used to determine whether performing this algorithm in reduced precision is wise for the particular application. We introduced the typical terminology that is used within these papers in Unit 5 and, as a case study, considered the Gaussian Elimination Method.

As well as considering the accuracy of the calculation when using a mixed-precision approach, there are code structures where the overheads from converting the data from one precision to another can be greater than the gains from using the reduced precision. In Unit 6, we considered different code structures and the affect they have on the execution times.

Prior to embarking on the job of converting a code to mixed-precision, considerations into whether the accuracy is acceptable and whether the underlying code structure is amenable should always be taken. However, there will always be occasions where trial and error will need to be used to see whether mixed-precision is suitable for one's application.