

---

# Floating-Point Arithmetic Exercises and Examples

---

**We will go through several examples of floating-point code which demonstrates some of the points made during the lectures**

- One of the openlab staff will describe which system you should use and in which directory you will find the exercises. Set your working directory to that location.
- The code in the exercises can be compiled with gcc 4.7.1 and icc 13.0.
- Source the setup script **Setup-for-exercises** and then verify that the correct compilers are available to you. You should see something like

```
-bash-4.1$ . ./Setup-for-exercises
The environment for Intel software tools for Linux has been set up:
License ports:
28518@lxlic01.cern.ch,28518@lxlic02.cern.ch,28518@lxlic03.cern.ch
-bash-4.1$ which gcc && gcc -v
/oplashare/sw/linux/x86_64/gcc/slc6/gcc-4.7.1/bin/gcc
Using built-in specs.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=/oplashare/sw/linux/x86_64/gcc/slc6/gcc-4.7.1/libexec/gcc/x86_64-unknown-linux-gnu/4.7.1/lto-wrapper
Target: x86_64-unknown-linux-gnu
Configured with: ../gcc-4.7.1/configure --enable-languages=c,c++,fortran
--prefix=/oplashare/sw/linux/x86_64/gcc/slc6/gcc-4.7.1
Thread model: posix
gcc version 4.7.1 (GCC)
-bash-4.1$ which icc && icc -v
/oplashare/sw/linux/x86_64/intel/xe2013/composer_xe_2013.0.079/bin/intel64/ic
c
icc version 13.0.0 (gcc version 4.7.0 compatibility)
```

- In those exercises which supply a makefile, define **CXX=icc** to use icc to build the project instead of the default compiler gcc. The default target is **all**: build all the files required in the exercise. There is also a **realclean** target to delete all the files created. NOTE: you must build the **realclean** target before switching compilers. A typical sequence might be

```
make # build with gcc
# work on the exercise
# save any files which you way want later; e.g., executables
make realclean # clean before switching compilers
make CXX=icc # build using icc
# work on the exercise
# save any files which you way want later
make CXX=icc realclean # remove icc-created files
```

## 01\_numeric\_limits

This program demonstrates use of the templated `numeric_limits` functions. The program prints the values of various significant floating-point quantities relevant to `float`, `double`, `long double` and `__float128` datatypes. Note that `__float128` is not considered a fundamental datatype by the GNU compiler system. Thus, the standard GNU C++ headers do not include instantiations specialized for `__float128`. They are supplied in `numeric_limits.hpp`. They may not be entirely correct.

Compare the outputs of the gcc and icc versions. They should be the same except for a few `long double` exceptional values. The differences appear to be the result of how "ignored" bits in the 80-bit storage format are handled by the two compilers. Note that the `long double` routines are part of the standard C++ header `limits`.

## 02\_decode

This sample program decodes the sign, exponent and significand fields of the `single`, `double` and `long double` floating-point data types. It demonstrates the use of C language system header files to map the various fields of the storage format of a floating-point number onto structures which can be used to examine (or modify) them.

## 03\_summation

This directory contains the files to build a program which sums a collection of double precision values in a number of different ways. The program also measures the execution time of each method.

The summation techniques used include

- simple summation, with and without sorting
- summation using increased precision
- an error free transformation (EFT) which implements a highly accurate summation scheme without use of increased precision
- vectorization
- unrolled loops
- OpenMP
- reduction methods from Intel® Threading Building Blocks (TBB)

First make some observations:

- Run the program several times. Build it with both gcc and icc.
- Are the results always the same? Are they the same for either compiler?
- Which techniques give the same result each time?
- Which give results which vary?
- Can you explain why?

- Examine the manner in which the pseudo-random numbers are generated. They aren't randomly distributed on  $[0,1)$ .
- Look at the source files containing the various functions. Understand how the various techniques are implemented.
- The main program demonstrates a method to measure elapsed time using OpenMP. Note that OpenMP is only used "computationally" in one of the summing routines.

Which of the techniques do you think provides the "right" value for the sum. What do you think "right" means in this case?

Examine the way in which the pseudo-random numbers are generated. Can you correlate that with the different results? E.g., compare the "unroll by 4" results with the "unroll by 5" results.

Identify the techniques which use multiple threads of execution or multiple partial sums. Why does this affect the results?

## 04\_dot\_product

This directory contains several prototype programs for calculating dot products. Examine each one and see how the various error free transformations discussed during the lecture are applied to the problem.

If you wish to be creative, create a main program which incorporates one or more of these routines and use it to calculate a dot product. Can you correlate your results with the condition number associated with the dot product?

## 05\_quadratic

This directory contains a very rudimentary program which solves the equation

$$ax^2 + bx + c = 0$$

for its real roots. It does it in a very numerically naïve way.

What roots do you find for  $a = 1$ ,  $b = 2000$  and  $c = -3$ ? Check their suitability by evaluating the equation directly.

Fill in this table (where  $x_1$  and  $x_2$  are the roots found):

$a$	$b$	$c$	$x_1$	$x_2$	$f(x_1)$	$f(x_2)$
1	2000	-3				
2	-4000	-1				
5	8000	+2				

---

Try to improve the program so that various “pathological” combinations of  $a$ ,  $b$  and  $c$  are properly handled. Look at Goldberg’s paper for some ideas. <http://perso.ens-lyon.fr/jean-michel.muller/goldberg.pdf> or [http://docs.oracle.com/cd/E19957-01/806-3568/ncg\\_goldberg.html](http://docs.oracle.com/cd/E19957-01/806-3568/ncg_goldberg.html).