Improving the accuracy of floating-point computations in *FullSWOF_1D*.

Frédéric Darboux and Carine Lucas
frederic.darboux@orleans.inra.fr carine.lucas@univ-orleans.fr

2012-08-13

When running the 8 benchmarks of *FullSWOF_1D* on several computers, we noticed differences in the results for two benchmarks: the bump at rest and the Thacker (others were strictly identical). Performing some further testing, we also uncover some problems of symmetry: for example, inverting the geometry of the dam break from left-to-right to right-to-left did more than simply inverted the results.

These differences originated from approximations in the floating-point computations, and these approximations depended on the computer. After a diagnosis phase, various changes were made, both in the code and in the compiler options. These changes are described below. This work leads us to formulate advices to users of *FullSWOF_1D*.

Testing & Changes

The first change we made was to replace each $x^2 - y^2$ by $(x - y)(x + y)$. This modification affected the main part of the numerical scheme (see `scheme.cpp`) for the time evolution of the discharge and some fluxes. It is well-known that this way of computing reduces floating-point errors. For more details, we refer to `http://www.codeproject.com/Articles/29637/Five-Tips-for-Floating-Point-Programming` and references therein.

In order to solve symmetry problems, we had to use long doubles in some parts of the code. Namely, a computation with long double is now done in *FullSWOF_1D* in `hll.cpp`, `muscl.cpp`, `scheme.cpp`. Moreover, the compilation option `-ffloat-store` was added.

Let us study the influence of these changes on the results. In the following, we make a distinction between a 64-bit computer under Mac OS X and a 32-bit computer under Linux Ubuntu. GNU C++ was used in both cases.

Only two out of the eight benchmarks differed depending on the float type and compilation option: the bump at rest and the Thacker. These two test cases are the only ones to involve very small perturbations of water height, and null velocities, and this may explain the difficulties to catch the analytic solution.

This study (see tables 1-2) shows that each long double is necessary to obtain results exactly identical to the reference solution. Moreover, it points out that the `-ffloat-store` option improves the result accuracy but roughly doubles the computation time. However, excepted for some specific points and output variables (especially the Froude number), most of differences were relatively small.

Conclusions & Recommendations

- There are still differences whether you run *FullSWOF_1D* under a 32-bit or a 64-bit operating system.
- The option `-ffloat-store` helps in obtaining accurate results for very small water heights. This is useful in order to compare the numerical results with analytic solutions. But if you want to compare the numerical results with experimental results, you can remove this option to save computation time, especially if very shallow flows are not involved. In particular, you should

keep in mind that this option is very time consuming and you must balance the accuracy versus the computation time.
- Other improvements could be done. Do not hesitate to suggest modifications!
- Currently, only the HLL and the MUSCL reconstruction have been tested and modified. Tests should also be done for the other fluxes and reconstructions.

Experiment number	Long double HLL	Long double MUSCL	Long double scheme	-ffloat -store	Difference with reference (bump at rest, Thacker)	Computation time (Mega clock ticks)
reference	yes	yes	yes	yes		860
1	no	yes	yes	yes	yes	670
2	no	yes	yes	no	yes =1	380
3	yes	no	yes	yes	no	780
4	yes	no	yes	no	no	420
5	yes	yes	no	yes	yes	740
6	yes	yes	no	no	yes =5	410
7	yes	yes	yes	no	no	415

Table 1: Mac - 64 bits

Experiment number	Long double HLL	Long double MUSCL	Long double scheme	-ffloat -store	Difference with reference (bump at rest, Thacker)	Computation time (Mega clock ticks)
reference	yes	yes	yes	yes		540
1	no	yes	yes	yes	yes	540
2	no	yes	yes	no	yes	270
3	yes	no	yes	yes	no	540
4	yes	no	yes	no	yes	270
5	yes	yes	no	yes	yes	530
6	yes	yes	no	no	yes	260
7	yes	yes	yes	no	yes =6	260

Table 2: Linux - 32 bits