GPUSPH Installation Guide

version 5.0 — June 2019

Contents

1 Introduction

GPUSPH is an implementation of Smoothed Particle Hydrodynamics (SPH) on NVIDIA CUDA-enabled graphics cards. The first version of GPUSPH was developed by Alexis Hérault, guided by SPHysics, and presented at the Third SPHERIC Workshop in Lausanne, Switzerland in 2008. The graphics processing unit (GPU) implementation came from GPU-LAVA, a lava flow program, developed by Hérault and Bilotta at INGV in Catania, Italy. The present version of GPUSPH is open source, licensed under the GNU General Public License (`www.gnu.org/licenses/gpl.txt`).

Smoothed Particle Hydrodynamics (SPH) is a Lagrangian meshless numerical method that was developed in astrophysics by Lucy (1977) and Gingold and Monaghan (1977). Its first application to free surface flows (e.g. dam breaks and waves) was by Monaghan et al. (1994). Since in SPH the interactions between particles involve many neighbors (several hundreds in three dimensions), it suffers from high computational costs. This motivated the development of massively parallel SPH codes, in particular codes running on graphics cards due to their performance and relatively low cost.

The development of sophisticated graphics cards is driven by the demands of advanced computer gaming, in particular to handle three-dimensional graphics for the computer display. Each of these graphics cards has numerous streaming processors to do the mathematics of image rotation, resizing etc. With the advent of the *CUDA* programing language from NVIDIA in 2007, simple C++ language can be used to access the mathematical power of these massively parallel cards. For computer simulations that are not data-intensive, GPU programming provides supercomputer capabilities at commodity prices.

Some timing information can be found in Hérault et al. (2010), showing that using the GPU is far faster (orders of magnitude) than using a CPU to compute SPH models. Speedups of 100 can be achieved for parts of the code when compared to serial versions of the code.

The first version of GPUSPH was running on NVIDIA's Compute Capability (CC) 1.x cards, GeForce 8xxx cards. From this first version we tested GPUSPH on all NVIDIA architectures (from Fermi CC 2.x to the latest Pascal CC 6.x). NVIDIA dropped support for CC 1.x and will soon do it for CC 2.x. So at the moment GPUSPH will run on any card with CC 2.x or higher (from Fermi up) but we expect to drop soon the support for CC 2.x. When done GPUSPH will run on any card with CC 3.x or higher (from Kelpler up).

This guide is divided into several sections. First, the installation and set-up of the GPUSPH code is explained and some example problems to illustrate its use are

provided. The second chapter goes through all the steps necessary to build a new simulation and post-process the results. The third chapter deals with an overview of SPH, with which the reader should have some familiarity. Finally we discuss the nature of the GPUSPH program in some detail.

2 Installation of GPUSPH

The first step to run GPUSPH is to install the NVIDIA company's CUDA compilers and libraries (directions given below). CUDA is an extension of the C++ language to allow C++ to talk to the graphics card.
The second step is to install the open source software, CHRONO, which simulates rigid body dynamics. This library is used for any rigid objects that move, such as floating objects or objects moved by fluid flow.
The third step is to obtain, compile and run GPUSPH.
Remark: to run multi-node simulations, you also need to install OpenMP (>= 1.8.4).

2.1 Installing CUDA

Ensure that your computer has an NVIDIA graphics card that is CUDA enabled. The NVIDIA website has a list of all the CUDA-enabled graphics cards: `www.nvidia.com/object/cuda_gpus.html`. You can check whether CUDA is already installed on your machine by launching the command:

```
nvidia-smi
```

from the terminal. If CUDA is installed it will give you information on the current state of the NVIDIA graphics card(s) on the machine.
Note: GPUSPH runs on cards with Compute Capabillity at least 2.0
Remark: regarding the choice of the GPU, the more CUDA cores and the more memory on the card, the better. Anyway a mid range mobile GPU's like GT750m/GT840 with 1 or 2GB of memory is sufficient to run significant simulations. A laptop with such a GPU will be a perfect mobile developing and testing platform.
The GPU programming language CUDA can be obtained from the NVIDIA website, CUDA Zone. The CUDA Toolkit and CUDA Software Development Kit (SDK) need to be installed for your operating system along with the video driver. These packages include the CUDA compiler `nvcc`, which is needed to develop executable code, and the graphics card driver that allows your program to access the GPU card.

Download the relevant driver for your machine from: `http://www.nvidia.com/Download/index.aspx?lang=en-us` and the CUDA toolkit from: `https://developer.nvidia.com/cuda-downloads` Follow the instructions provided by NVIDIA for the installation.
To ensure that all is installed correctly and working, you should compile and run the SDK examples, which include many programs that illustrate the capabilities of CUDA and the GPU; for example, NVIDIA's sorting program `radixSort` is used by GPUSPH to organize the neighbor list. Some interesting SDK programs are `fluidsGL` and `particles`. To compile the SDK programs, after the SDK is installed, go to `/Developer/GPUComputing/C` and (on a unix/linux or mac machine), type `make` on a terminal window command line. This should create a directory of executable examples located within the C directory called bin/darwin/release for the mac and bin/linux/release for a linux machine. In this directory, type `./fluidsGL` to run the `fluidsGL` example. You should see a green window open on your desktop. Use the mouse to stir up the fluid. The example program Particles is worth playing with as well, as it provided a basis for developing GPUSPH.

2.2 Installing GPUSPH

The GPUSPH source code is hosted on GitHub. The project's GitHub page is `http://github.com/GPUSPH/gpusph`.
To obtain the GPUSPH code, you can either use the `git` revision control system, or download a `.zip`ped archive of a specific version. This manual refers to version 5.0 of GPUSPH.
If you have `git` installed, you can use

```
git clone https://github.com/GPUSPH/gpusph.git
cd gpusph
git checkout v5.0
```

to get version 5.0 specifically. Otherwise, download the `.zip`ped archive from `http://github.com/GPUSPH/gpusph/archive/v5.0.zip`, and then

```
unzip v5.0.zip
cd gpusph-5.0
```

(you may remove `v5.0.zip` afterwards).
Within the top directory, you can find the `Makefile`, a `src` directory (holding the main GPUSPH source), a `scripts` directory (holding various auxiliary scripts), a copy of the license, settings to produce internal documentation with Doxygen, and a sample Digital Elevation Model (DEM) data file.

The most interesting source files in `src` are the `Problem`s. A few sample problems are shipped with GPUSPH, showing how to employ specific features. You can get a list of the available problems by running

```
make list-problems
```

To build and test GPUSPH, you can run

```
make test
```

which should automatically detect your configuration, such as the compute capability of your GPU as well as the availability of optional libraries such as MPI (for mulit-node support) or HDF5 (to read HDF5SPH data files).
When the building completes, you will have some new directoryes (`build` and `dist`) and a `GPUSPH` soft link to the compiled binary. `make test` will also automatically run `./GPUSPH` for you.
After building, simply runnning `./GPUSPH` will run the program again.

2.3 Installing the CHRONO library

The CHRONO website provides information for how to install CHRONO: `http://api.chrono.projectchrono.org/tutorial_install_chrono.html`
Remark: There is no need for the Irrlicht library with GPUSPH.
In this section we summarize the steps for the CHRONO library installation. To install CHRONO, besides the GPUSPH requirements you need to have `cmake` installed and a cmake interface like `ccmake` on Linux.
First, create a directory where to install CHRONO:

```
mkdir install_chrono
```

In that directory, clone the CHRONO repository from Github in a source directory:

```
cd install_chrono
git clone https://github.com/projectchrono/chrono.git source
```

This command will download the CHRONO repository in a folder named `source`.
Create a folder where to build CHRONO:

```
mkdir build
```

From the `build` repository, run `cmake`:

```
cmake ../source
```

Configure the compilation options with `ccmake`:

```
ccmake .
```

set the following options to off:

```
ENABLE_MODULE_CASCADE              OFF
ENABLE_MODULE_COSIMULATION         OFF
ENABLE_MODULE_FEA                  OFF
ENABLE_MODULE_FSI                  OFF
ENABLE_MODULE_IRRLICHT             OFF
ENABLE_MODULE_MATLAB               OFF
ENABLE_MODULE_MKL                  OFF
ENABLE_MODULE_OPENGL               OFF
ENABLE_MODULE_PARALLEL             OFF
ENABLE_MODULE_POSTPROCESS          OFF
ENABLE_MODULE_PYTHON               OFF
ENABLE_MODULE_VEHICLE              OFF
ENABLE_OPENMP                      OFF
```

Once this is done, you can compile the CHRONO project (still from the `build` folder):

```
make
```

and install the library (also from the `build` folder):

```
sudo make install
```

3 Choosing the GPUSPH `Problem` and other compilation options

You can test a different problem by using:

```
make OtherProblem test
```

where `OtherProblem` is the name of a different problem. You can get a list of available problems with `make list-problems`.
There are a number of other options available. A complete list of the options and their description can be obtained by running `make help-options`. All options (with the exception of `plain` and `echo`) are persistent across compilations, so they can be set once with `make option=value`, and subsequent executions of `make` will remember the `value` set.
The `make` options are listed below:

- `target_arch` - if set to 32, force compilation for 32 bit architecture
- `problem` - Name of the problem. Since version 5, you can just use the problem name as a target to build that particular test case.
- `dbg` - 0 no debugging, 1 enable debugging
- `compute` - 11, 12, 13, 20, 21, 30, 35, etc: compute capability to compile for (default: autodetect)
- `fastmath` - Enable or disable fastmath. Default: 0 (disabled)
- `mpi` - 0 do not use MPI (no multi-node support), 1 use MPI (enable multi-node support). Default: autodetect
- `hdf5` - 0 do not use HDF5, 1 use HDF5, 2 use HDF5 and HDF5 requires MPI. Default: autodetect
- `verbose` - 0 quiet compiler, 1 ptx assembler, 2 all warnings
- `plain` - 0 fancy line-recycling stage announce, 1 plain multi-line stage announce
- `echo` - 0 silent, 1 show commands
- `chrono` - 0 do not use the CHRONO library, 1 use the CHRONO library

To view your current make options type `make show` instead of make.

4 Example Problems

Simulations in GPUSPH are defined in terms of `Problem`s. Some example problems are provided with GPUSPH itself, to illustrate the basics of problem design, and how to use the fundamental building blocks provided by GPUSPH. Such building blocks include a variety of geometrical shapes to describe the (fixed) solid boundaries of the domain, as well as a number of objects that move following prescribed laws, such as gates, pistons and paddles.
These objects are designed to offer great flexibility in their use, far beyond what is shown in the sample problems. This flexibility should allow you to create very complex simulations by combining the objects appropriately.
The number of particles used in the test problems is deliberately taken as a small number, simply to allow for fast execution times even on older hardware. One of the

first tests to try is to increase the resolution by reducing the size of the particles. For example, by reducing the particle size from the default of 0.025m to the smalle 0.02m, `DamBreak3D` would run with $21,252$ particles instead of the default $10,664$.
This can be done in two ways. A permanent change comes about by editing the problem file (e.g. `DamBreak3D.cc`) and changing the value passed as argument of `set_deltap()` (e.g., replace `set_deltap(0.025f);` with `set_deltap(0.02f);`. The second way is to specify the particle size at runtime using the appropriate command line option (described below): e.g. `./GPUSPH --deltap 0.02`.

4.1 DamBreak3D

`DamBreak3D` is a case originally used by Gomez-Gesteira and Dalrymple (2004) for testing a prototype version of SPHysics. It is based on some experiments done by Árnason (2005) at the University of Washington. We assume an instantaneous breaking dam and the resulting flow impinging onto a rectangular object. The whole problem is contained within a bounding box, which extends 1.6m in length (x axis), 0.67m in width (y axis), and 0.4m in height. This is the experimental box. The fluid behind the dam is a rectangular box of water at one end of the tank at time equal to zero. The dam is assumed to break instantaneously so that the column of water, confined on three sides, collapses into the tank. In the tank there is a vertical rectangular object – the collapsing water column impacts on the tank and then flows up the front face of the object and around the sides. Finally the water hits the back wall of the tank. A screenshot of the simulation at time $0.6s$ is provided in the Figure 1.

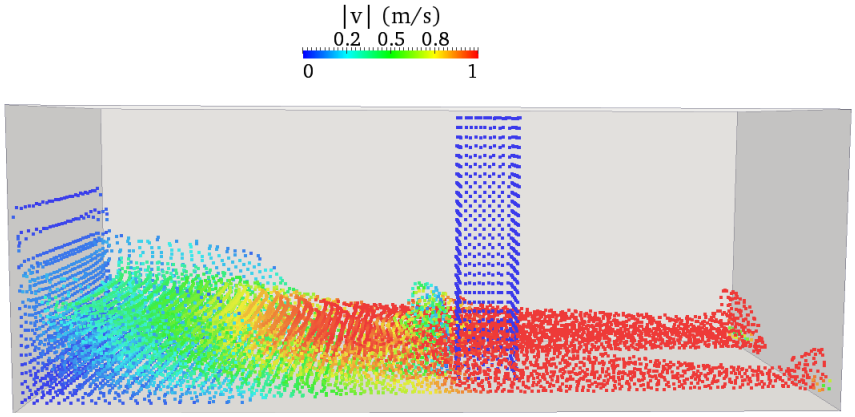

Figure 1: Screenshot of the DamBreak3D simulation at time $0.6s$.

4.2 DamBreakGate

In most laboratory experiments of dam breaks, the dam takes a certain amount of time to move out of the way. The example problem `DamBreakGate` illustrates the use of moving boundaries with prescribed motions. The problem is set up the same way as the `DamBreak3D` case, but there is a moving gate that is raised vertically with a linearly varying velocity. In this case, the gate will move with a velocity that is zero when the problem starts and that linearly increases with time until the gate is outside the domain. The effect on the dam break is that the escaping water is affected by the gage motion. (See Crespo et al. (2008)'s SPH modeling of Jánosi et al. (2004)'s experiment, where a moving gate was important.)
The moving gate is created by defining its geometry with a geometry type `GT_MOVING_BODY`, and overriding the Problem `moving_bodies_callback` function to determine the linear and angular velocity of the body.
A screenshot of the simulation at time $0.8s$ is provided in the Figure 2.

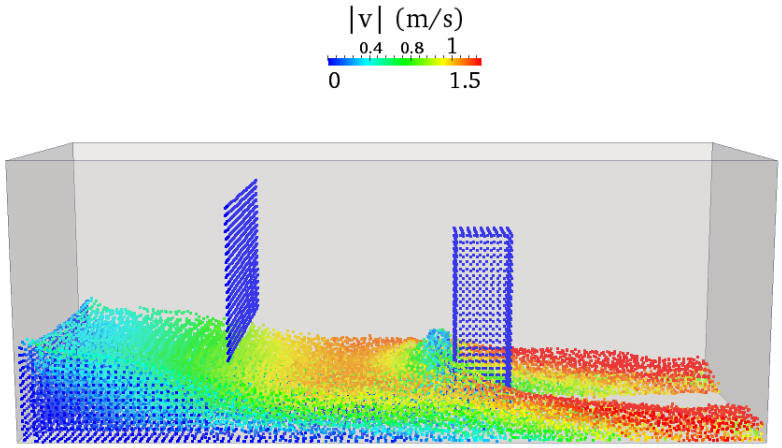

Figure 2: Screenshot of the DamBreakGate simulation at time $0.8s$.

4.3 OpenChannel

This problem represents an instantaneous start up of a highly viscous end dense fluid flow in an open channel on a 9 deg slope. The channel is rectangular in cross-section (1m wide and 0.7m deep) and the computed length of the infinitely long channel is 2m. The side walls are fixed (Leonard-Jones boundary force) while the computational

ends of the domain are periodic, so that a particle leaving the downstream end of the model domain enters the upstream end at the same place, 2m upstream.
The periodic boundary here is used in the x direction, although boundaries in other problems can be periodic in the other directions as well. The key parameter in the problem statement is the simulation framework `periodicity`, which can be set to any combination of `PERIODIC_X`, `PERIODIC_Y`, `PERIODIC_Z` to indicate peridocity along each of the axes. Figure 3 shows the shape of the velocity field in the channel after time convergence.

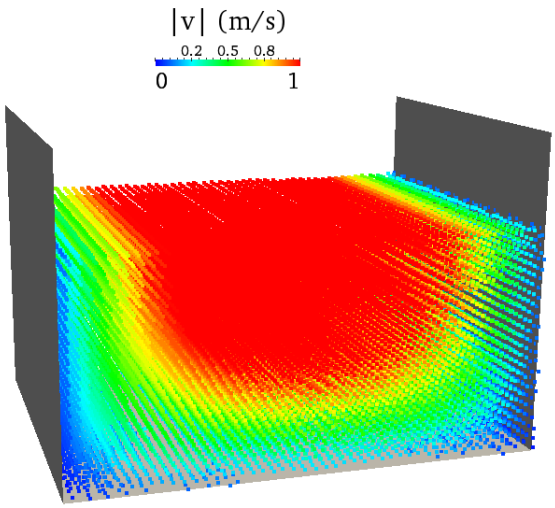

Figure 3: Screenshot of the OpenChannel simulation after time convergence.

4.4 WaveTank

WaveTank uses a moving boundary to create a paddle wavemaker at one end of a wave tank with a sloping bottom (bottom slope is 4.2364 deg). The wavemaker motion is controlled by the `moving_bodies_callback` function. In this case, the length of the paddle is 1.0m and the paddle pivots about an origin `m_origin`; here, the pivot is located 0.1344m below the bottom and 0.13m from the front wall of the tank. To specify the paddle motion, the angular frequency of the motion ($2\pi/T$, where $T = 1$s is the wave period), and the wave paddle stroke at the water surface ($S = 0.1$m) are given in the variables `mb_omega` and `mb_amplitude`. To change the

stroke and the frequency of the wave paddle, you must change these variables in the problem file, `WaveTank.cc`. Figure 4 shows a screenshot of the simulation at time $9s$.

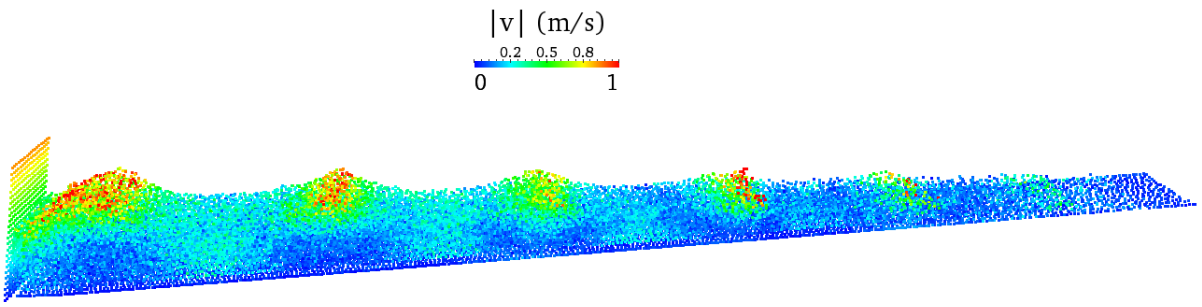

Figure 4: Screenshot of the WaveTank simulation at time $9s$.

4.5 SolitaryWave

SolitaryWave is similar in set up to the WaveTank example, except that a piston moving boundary is used. The motion of a vertical plate is determined by the method of Goring (1979), available in PDF format from: `http://caltechkhr.library.caltech.edu/50/` The full excursion (stroke) of the paddle is the variable `S`.

4.6 Seiche

The Seiche problem is to examine the influence of shaking on a rectangular container of size: $\ell = 0.707$m, $w = \ell/2$, and depth, $H = 0.5$m. The purpose of the example is to illustrate the ability to vary gravity in a problem. As the problem starts, there is water in the container. After 0.3s, gravity is modified by adding a component in the x direction, such that the total gravity vector is `set_gravity(3.*sin(9.8*(t-m_gtstart)), 0.0, -9` which means that the container is shaken with a sinusoidal motion with angular frequency of $9.8s^{-1}$ (period $= 0.64$s), with a magnitude of 3m/s^2 until time `m_gtend=3.0` is reached, when the gravity vector once again returns to the vertical acceleration of gravity. After this time, the seiching motion starts to decrease in amplitude.
The variation of gravity with time (and any stop (`m_gtend`) and start times) is prescribed in a user-supplied (in the problem) `g_callback` function.

4.7 DEMExample

This is an example showing how to use GPUSPH's support for Digital Elevation Models (DEMs). It loads the topography of the bottom of the domain from a file called `half_wave0.1m.txt`, shipped with GPUSPH. A different DEM can be used, by either changing the name in the source `DEMExample.cu` file, or by providing the new name as argument to the `--dem` command-line option to GPUSPH.

5 GPUSPH Command Line Options

When running from the command line, there are several options available to you to alter some aspects of the GPUSPH run.

--device *integer* Choose which GPU(s) to use for the run. On the command line: `./GPUSPH --device N`, where N is the (integer) number of the device you wish to use. To find the number associated with each of your CUDA-enabled devices (graphics cards), you can use the CUDA SDK program DeviceQueryDrv. If you only have one CUDA-enabled GPU, the only possible choice for N is 0, which is the default. If you want to run the simulation on several GPUs, the command is: `./GPUSPH --device i,j,k`

--deltap *float* Change the resolution (inter-particle spacing) at which the problem should be run.

--tend *float* The model time in seconds when you wish the model to stop.

--dt *float* Force the use of the fixed, specified time-step, even if the problem would use dynamic time-stepping otherwise.

--dem *string* For the Problem DEMExample: the name of the DEM file to use.

--resume *fname* Resume from the given file (HotStart file saved by HotWriter).

--checkpoint-every *float* HotStart checkpoints will be created every VAL seconds of simulated time (float VAL, 0 disables).

--checkpoints *integer* Number of HotStart checkpoints to keep.

--maxiter *integer* Break after this many iterations.

--dir *string* Use given directory for dumps instead of date-based one.

--nosave Disable all file dumps but the last.

--gpudirect Enable GPUDirect for RDMA (requires a CUDA-aware MPI library).

--striping Enable computation/transfer overlap in multi-GPU (usually convenient for 3+ devices).

--asyncmpi Enable asynchronous network transfers (requires GPUDirect and 1 process per device).

--num-hosts *integer* Uses multiple processes per node by specifying the number of nodes.

--byslot-scheduling MPI scheduler is filling hosts first, as opposite to round robin scheduling.

--debug *flags* Enable specified debug flags.

--help Show the help and exit.

5.1 Debugging GPUSPH execution

To assist developers when introducing new GPUSPH features, it is possible to trace the execution of the simulator, providing insight on the evolution of some of the internal structures using the provided debug flags. Multiple debug flags can be enabled in a single run, by passing a comma-separated list as an argument to `--debug`. For example:

```
./GPUSPH --debug print_step,inspect_buffer_access
```

The `validate_init_position` debug option is also useful for users that want to further validate their initial setup.
The debug flags currently supported are:

print_step print each step and command as it is being executed;

neibs debug the neighbors list on host, saving it to file;

forces debug forces on host, saving them to file;

numerical_density debug relative density variation on host, saving it to file;

inspect_preforce inspect pre-force particle status; this saves the particle system status before each force computation;

inspect_pregamma inspect pre-gamma integration particle status; this saves the particle system status before the gamma integration kernel execution;

inspect_buffer_access inspect buffer access; show how each buffer is being accessed (reading or writing, and from which state) during execution of each command; note that this needs compile-time support, by enabling *#define DEBUG_BUFFER_ACCESS 1* in `src/buffer.h`;

inspect_buffer_lists inspect the state of the particle system and its buffer lists (states and free buffer pool) after each command;

check_buffer_update check buffer update; this is used to check that all and only modified buffers are exchanged in an `UPDATE_EXTERNAL` call;

check_buffer_consistency check buffer consistency; when this is enabled, after every command GPUSPH will verify that the shared parts of the subdomains in multi-GPU simulations are consistent across devices; note this needs compile-time support, by adding `-DINSPECT_DEVICE_MEMORY` to `CPPFLAGS` in `Makefile.local`;

clobber_invalid_buffers clobber invalid buffers; when this is true, every time a buffer is marked invalid, its content will be clobbered (reset to the initial value, typically NAN or equivalent); useful to check that stale data is not being used inadvertently;. note that this needs compile-time support, by enabling *#define DEBUG_BUFFER_ACCESS 1* in `src/buffer.h`;

validate_init_positions throw an exception (instead of just warning) if a particle is out of bounds during initialization of the problem;

benchmark_command_runtimes measure (and show) command runtimes.

6 Running multi-node simulations

GPUSPH can distribute the computation of a simulation on multiple GPU devices attached to different nodes of a cluster in different ways.
Say we want to launch a simulation on N nodes, each with D devices (with CUDA device numbers ranging from 0 to $D-1$); the total number of devices in the simulation will be $N \times D$. We can run either:

- one process per node, D GPUs per process

- 2 processes per node, $D/2$ GPUs per process
- 4 processes per node, $D/4$ GPUs per process
- etc.

Additionally, some MPI implementations have built-in support for CUDA, which allows for faster communication between devices on different nodes. Experimental support for this feature can be enabled in GPUSPH with the `--gpudirect` command-line option.
The best decision on how to distribute the computation across nodes and devices depends on the queue policy of the cluster, on the network topology, on simple a posteriori performance tests, on the capabilities of the MPI implementation, etc.
If we wanted to run the simulation on all the devices of one node, we would run in an interactive shell:

```
./GPUSPH --device 0,1,...D-1
```

Running the same simulation on multiple nodes only requires to run the same command within the reference MPI launcher (usually a script called `mpirun`); GPUSPH will retrieve the necessary information about the launch environment directly from the MPI runtime and will organize the node-to-node communication accordingly.

```
mpirun -np N ./GPUSPH --device 0,1,...D-1
```

This command leaves to MPI the choice of which nodes to use in the network, if more than N are available. It is always safe to provide MPI a list of hostnames corresponding to the nodes chosen to run the simulation. If the file containing the list of hostnames is called `myhostsfile`, a typical syntax will be:

```
mpirun -np N -hostfile ./myhostsfile \
./GPUSPH --device 0,1,...D-1
```

Please note the syntax may vary from one MPI library to another. For example, MVAPICH uses `-hostfile` while OpenMPI `--hostfile`.
Let us now see the command line options needed to run the same simulation with more processes (and thus on more nodes) and a smaller number of devices per process. In this case we need to inform both the MPI runtime and GPUSPH. For the former, we simply decrease the number of processes to start (with the `-np` option); for the latter, we need to shorten appropriately the list of devices passed with the `--device` option. If our aim is to run $N * 2$ processes each using $D/2$ devices, we then run:

```
mpirun -np N*2 -hostfile ./myhostsfile \
./GPUSPH --device 0,1,...D/2-1
```

Here we need to take care of a few important details. If the list of available hosts contains at least $N * 2$ hostnames, the MPI runtime will start every process on a different physical node. But what happens if the list is shorter (e.g. N hosts only), or if we want anyway to use a smaller number of nodes (for example because part of the cluster is already used by other processes)? The MPI runtime will start multiple processes per node and the GPU device numbers will be likely to conflict (i.e. two different processes might try to use the same GPU device for different parts of the simulation domain, causing a performance slowdown or a failure if the CUDA devices are set in "exclusive mode"). In this case, we must inform GPUSPH that the number of physical hosts (i.e. nodes) is smaller than the number of processes, so that it will shift the GPU device numbers of the appropriate processes and no GPU device will be accessed by two processes. The corresponding option is `--num-hosts`:

```
mpirun -np N*2 -hostfile ./myhostsfile \
--num-hosts N ./GPUSPH --device 0,1,...D/2-1
```

But there is another important detail. There are different ways the MPI runtime can distribute the processes across the nodes. Two very common policies are "by slot" (fill-first) and "by node" (round robin). The scheduling policy affects the association between the process ranks and the CUDA device numbers, so GPUSPH must be informed about it to use the appropriate offsets. GPUSPH assumes a round robin schedule is being used; if this is not true, the `--byslot-scheduling` option must be passed:

```
mpirun -np N*2 -hostfile ./myhostsfile \
--num-hosts N --byslot-scheduling \
./GPUSPH --device 0,1,...D/2-1
```

There is no optimal policy in general as its performance depends on the node load and the node-to-node connection speed. It is worth trying both to check whether one is more performant than the other. The default policy is usually "by slot"(fill-first, usually preferred by non GPU-based softwares) but it is always safer to explicitly set it for every run. In OpenMPI the corresponding options are `--byslot` and `--bynode`.
One final possibility is to run one process per device. This is the most consuming option from the point of view of the host memory, since every process will allocate on host the whole simulation scenario, but it might be useful if the GPUDirect feature is being used but the MPI runtime does not support multiple devices per process, or if the amount of data to be saved on file is large or the saving frequency is very high, so that saving would benefit from a parallel dump (every process saves its part of the simulation independently from the other processes).
Finally, let's see some practical examples. Suppose our cluster has 12 nodes, each

equipped with 4 GPU devices. We need to run a simulation on 12 devices. To run 3 processes on 3 hosts, each using 4 devices, we will run:

```
mpirun -np 3 -hostfile ./myhostsfile ./GPUSPH --device 0,1,2,3
```

To run 6 processes on 6 hosts, each using 2 devices, we will run:

```
mpirun -np 6 -hostfile ./myhostsfile ./GPUSPH --device 0,1
```

Note that another simulation can be run simultaneously on the remaining 2 devices of the same nodes, with:

```
mpirun -np 6 -hostfile ./myhostsfile ./GPUSPH --device 2,3
```

To run 6 processes on 3 hosts, each using 2 devices, we will run:

```
mpirun -np 6 -hostfile ./myhostsfile_with3hosts \
--num-hosts 3 ./GPUSPH --device 0,1
```

(–byslot-scheduling might also be necessary)
To run 12 processes on 12 hosts, each using 1 device, we will run:

```
mpirun -np 12 -hostfile ./myhostsfile ./GPUSPH --device 0
```

Note that another simulation can be run simultaneously on one of the free devices of each node (e.g. device number 3), with:

```
mpirun -np 12 -hostfile ./myhostsfile ./GPUSPH --device 3
```

Please note the options for the MPI library always precede the GPUSPH executable name. If the MPI library supports it, we also suggest enabling the option to tag each line of the output with the process rank that has generated it; in OpenMPI, the option is called `--tag-output` while in MVAPICH `-prepend-rank`. This will come very helpful when the logs need to be analyzed (tip: use grep to separate the logs if they are multiplexed).
If you need to use any queue-management system, remember to inform it about the desired topology, coherently with the options passed to GPUSPH. For example, with PBS you would set the `nodes` and `ppn` parameters for the number of hosts and processes per host, respectively.

7 Installing pre/post processing tools

7.1 Installing SALOME

You can download SALOME from: `http://www.salome-platform.org/downloads/current-version`

For this you need to register on the SALOME website. Then, follow the installation instructions from the SALOME website and the installer.
Remark: SALOME is used for SA boundary pre-processing, to generate an STL mesh of the boundaries. You can of course use another mesher of your choice for this step.

7.2 Installing CRIXUS

Crixus is a preprocessing tool for GPUSPH.
Prerequisites:

- cmake $\geq$ 2.8
- cuda
- hdf5 $\geq$ 1.8.7

Getting CRIXUS:

```
git clone https://github.com/Azrael3000/Crixus.git
cd Crixus.git
```

Compiling Crixus: Crixus uses CMake for compilation. Let us assume that you have CRIXUS in a `Crixus.git` directory. and you want the building to happen in `Crixus.git/build` then follow the commands below:

```
mkdir build
cd build
cmake ..
make
```

Note that you should not run cmake in the main Crixus folder.
The binary is then located at `Crixus.git/build/bin/Release/Crixus`. Note that `make install` is not supported yet. To easily change the parameters of cmake you can use ccmake instead.
If hdf5 cannot be found due to lacking environmental variable you can edit the main CMakeLists.txt which has a commented line that reads:

```
#set(ENV{HDF5_ROOT} "/your/path/to/hdf5")
```

Uncomment it and set the respective hdf5 path in order to use your custom installation.
To finish the installation it is recommended to add the path to the CRIXUS binary to your `PATH` environment variable. Add this line in `~/.bashrc`:

```
export PATH=/your_path/Crixus.git/build/bin/Release/Crixus:$PATH
```

where `/your_path` is your path to the CRIXUS directory.

7.3 Installing PARAVIEW

PARAVIEW is directly available from the Linux packages.

Appendices

A GNU General Public License

GNU GENERAL PUBLIC LICENSE
Version 3, 29 June 2007

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program--to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same

freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

"This License" refers to version 3 of the GNU General Public License.

"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

"The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.

To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.

A "covered work" means either the unmodified Program or a work based on the Program.

To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work.

A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid

circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

b) The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in section 4 to "keep intact all notices".

c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts,

regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an "aggregate" if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical

medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A "User Product" is either (1) a "consumer product", which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular

product received by a particular user, "normally used" refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

"Installation Information" for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in

source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

d) Limiting the use for publicity purposes of names of licensors or authors of the material; or

e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your

license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An "entity transaction" is a transaction transferring control of an

organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A "contributor" is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's "contributor version".

A contributor's "essential patent claims" are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, "control" includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a "patent license" is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To "grant" such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. "Knowingly relying" means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is "discriminatory" if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory

patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others' Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to

address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE

USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

> <one line to give the program's name and a brief idea of what it does.>
> Copyright (C) <year> <name of author>
>
> This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.
>
> This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of

    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program. If not, see <http://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.

  If the program does terminal interaction, make it output a short
notice like this when it starts in an interactive mode:

    <program> Copyright (C) <year> <name of author>
    This program comes with ABSOLUTELY NO WARRANTY; for details type ‘show w’.
    This is free software, and you are welcome to redistribute it
    under certain conditions; type ‘show c’ for details.

The hypothetical commands ‘show w’ and ‘show c’ should show the appropriate
parts of the General Public License. Of course, your program’s commands
might be different; for a GUI interface, you would use an "about box".

  You should also get your employer (if you work as a programmer) or school,
if any, to sign a "copyright disclaimer" for the program, if necessary.
For more information on this, and how to apply and follow the GNU GPL, see
<http://www.gnu.org/licenses/>.

  The GNU General Public License does not permit incorporating your program
into proprietary programs. If your program is a subroutine library, you
may consider it more useful to permit linking proprietary applications with
the library. If this is what you want to do, use the GNU Lesser General
Public License instead of this License. But first, please read
<http://www.gnu.org/philosophy/why-not-lgpl.html>.

References

A. Crespo, M. Gómez-Gesteira, and R. Dalrymple. Modeling Dam Break Behavior over a Wet Bed by a SPH Technique. *Journal of Waterway, Port, Coastal, and Ocean Engineering*, 134(6):313–320, 2008. ISSN 0733-950X. doi: 10.1061/(ASCE)0733-950X(2008)134:6(313). URL http://ascelibrary.org/doi/abs/10.1061/%28ASCE%290733-950X%282008%29134%3A6%28313%29.

R. A. Gingold and J. J. Monaghan. Smoothed particle hydrodynamics - Theory and application to non-spherical stars. *Monthly Notices of the Royal Astronomical Society*, 181:375–389, November 1977.

Moncho Gomez-Gesteira and Robert Anthony Dalrymple. Using a Three-Dimensional Smoothed Particle Hydrodynamics Method for Wave Impact on a Tall Structure. *Journal of Waterway, Port, Coastal, and Ocean Engineering*, 130(2):63–69, 2004. ISSN 0733-950X. doi: 10.1061/(ASCE)0733-950X(2004)130:2(63). URL http://dx.doi.org/10.1061/(ASCE)0733-950X(2004)130:2(63).

Derek Garard Goring. *Tsunamis – The propagation of long waves onto a shelf*. PhD, California Institute of Technology, Pasadena, California, 1979.

Alexis Hérault, Giuseppe Bilotta, and Robert Anthony Dalrymple. SPH on GPU with CUDA. *Journal of Hydraulic Research*, 48(Extra Issue):74–79, 2010.

Imre M. Jánosi, Dominique Jan, K. Gábor Szabó, and Tamás Tél. Turbulent drag reduction in dam-break flows. *Experiments in Fluids*, 37(2):219–229, August 2004. ISSN 0723-4864, 1432-1114. doi: 10.1007/s00348-004-0804-4. URL http://link.springer.com/article/10.1007/s00348-004-0804-4.

L. B. Lucy. A numerical approach to the testing of the fission hypothesis. *The Astronomical Journal*, 82:1013–1024, December 1977. doi: 10.1086/112164.

J. J. Monaghan, P. J. Bicknell, and R. J. Humble. Volcanoes, Tsunamis and the demise of the Minoans. *Physica D: Nonlinear Phenomena*, 77(13):217–228, October 1994. ISSN 0167-2789. doi: 10.1016/0167-2789(94)90135-X. URL http://www.sciencedirect.com/science/article/pii/016727899490135X.

Halldór Árnason. *Interactions between an incident bore and a free-standing coastal structure*. PhD, UMI Dissertation Services, University of Washington, 2005.