DualSPHysics-Chrono

Index of contents

List of authors

Iván Martínez Estévez (ivan.martinez.estevez@uvigo.es). Universidade de Vigo, Spain.

Dr José M. Domínguez (jmdominguez@uvigo.es). Universidade de Vigo, Spain.

Dr Ricardo Canelas (ricardo.canelas@bentley.com). Bentley Systems, Lisbon, Portugal.

Dr Bonaventura Tagliafierro (btagliafierro@gmail.com). University of Salerno, Italy.

Dr Orlando García Feal (orlando@uvigo.es). Universidade de Vigo, Spain.

Professor Alejandro J.C. Crespo (alexbexe@uvigo.es). Universidade de Vigo, Spain.

Professor Moncho Gómez Gesteira (mggesteira@uvigo.es). Universidade de Vigo, Spain.

Description

This project contains the source code, dependencies and examples of **DualSPHysics v5.0.231** coupled to **Project Chrono v4.0.0**. This coupling is done via a communication interface (the so-called **DSPHChronoLib**). DualSPHysics and DSPHChronoLib are open-source codes released under GNU Lesser General Public License (LGPL). Project Chrono is an open-source code released under BSD-3-Clause License. This package is provided with the manuscript titled "**Coupling of an SPH-based solver with a multiphysics library**".

Structure of the package

Figure 1 shows a schematic of the resources provided in this package.

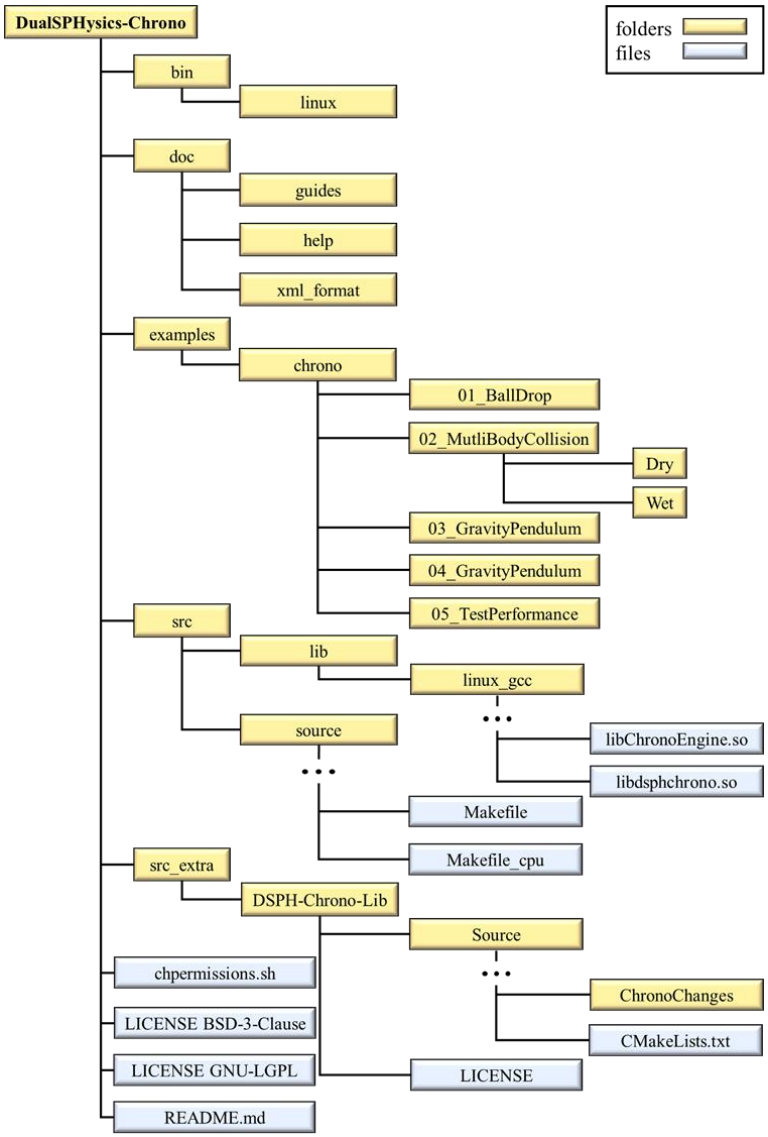


***Figure 1**. Schematic of the structure of the DualSPHysics-Chrono package*

The most relevant folders and files are detailed below:

- **bin/linux**: The binaries.
- **doc**
  - **guides**: Documentation to use the features of the code.
  - **help**: Documentation to execute DualSPHysics, the pre- and post-processing tools.
  - **xml_format**: XML templates to help the configuration of the cases.
- **examples**: Predefined examples to launch simulations.
- **src**
  - **lib/linux_gcc**: The libraries used by DualSPHysics. In this folder, both the **libChronoEngine.so** (Project Chrono) and the **libdsphchrono.so** (DSPHChronoLib) are included.
  - **source**: The source code of DualSPHysics. In this folder the *Makefiles* are included in order to compile the code.
- **src_extra/DSPH-Chrono-Lib**: Contains the resources of the DSPHChronoLib interface.
  - **ChronoChanges**: Source code files of Project Chrono, which contain modifications to allow the coupling.
  - **Source**: The source code of DSPHChronoLib. This folder includes a **CMakeLists.txt** file to build the project with the software **CMake**.

Compilation

In this section, the process to compile the three software tools involved in this work is explained.

Project Chrono

The source code of **Project Chrono** can be downloaded freely from its GitHub repository: chrono-4.0.0. Note that, the **Project Chrono** version 4.0.0 was used in the coupling with **DualSPHysics**. Some source files of **chrono-4.0.0** code were modified to add new functionalities in order to allow this coupling. These modifications are in the folder **DualSPHysics-Chrono/src_extra/DSPHChronoLib/Source/Chrono-Changes**. Therefore, copy the content of this folder into the downloaded package **chrono-4.0.0**. This action will replace the original files of Chrono.

Project Chrono can be compiled for Linux using CMake (https://cmake.org). CMake is a cross-platform and an independent building system for compilation. This software generates native building files (like *Makefile*) for any platform. The location of dependencies and the needed flags are automatically determined.

The building system needs the following dependencies:

- **CMake** version 3.0.0 or higher (https://cmake.org).
- GNU **G++** compiler 7.3.1 version or higher.
- **Make** tool for managing dependencies.
- File "**CMakeLists.txt**" in **chrono-4.0.0**.

Open CMake (cmake-gui). Paste the **chrono-4.0.0** folder path into the textbox labelled as ***Where is the source code*** and paste the build folder path into the ***Where to build the binaries textbox***. Once the paths are introduced, the `Configure` button should be pressed. A new dialog will appear asking for the compiler to be used in the project. Please remember that **only GNU G++ compiler 7.3.1 version or higher for 64 bit are supported**.

Introduce the path where **Project Chrono** will install the CMake properties of the project in **CMAKE_INSTALL_PREFIX**. The `Configure` button should be pressed again.

Press on `Configure` again. The output text should be ***Configuring done***. If the configuration was successful, now press the `Generate` button. This will generate a ***Makefile*** to compile the source files into the build directory and the output text should be ***Generating done***.

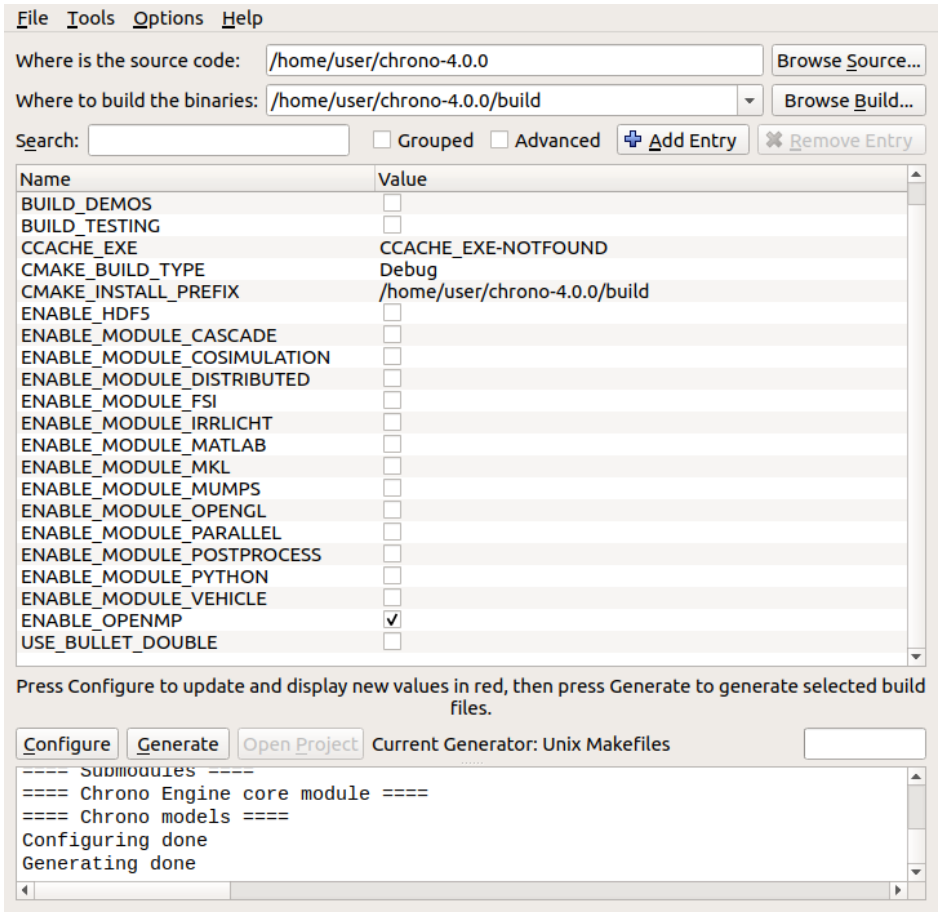


***Figure 2**. Building Project Chrono with CMake*

Open a Linux terminal and go into **chrono-4.0.0/build** folder and execute the command **make** on the terminal to compile the source.

```
$ cd chrono-4.0.0/build
$ make
```

If the compilation was successful, it will generate the **libChronoEngine.so** file into **chrono-4.0.0/build/lib**. Paste it into **DualSPHysics-Chrono/bin/linux** and into **DualSPHysics-Chrono/src/lib/linux_gcc**.

DSPHChronoLib

DSPHChronoLib is a dynamic library, which contains the source code that allows the communication between **DualSPHysics** and **Project Chrono**. The library **DSPHChronoLib** uses the CMake configuration generated by **chrono-4.0.0** build process. Therefore, building and compiling **chrono-4.0.0** first is mandatory in order to compile **DSPHChronoLib**. The resources of this commination interface are located in **DualSPHysics-Chrono/src_extra/DSPH-Chrono-Lib/Source**.

DSPHChronoLib can be compiled for Linux using CMake (https://cmake.org). CMake is a cross-platform and an independent building system for compilation. This software generates native building files (like *Makefile*) for any platform. The location of dependencies and the needed flags are automatically determined.

The building system needs the following dependencies:

- **CMake** version 3.0.0 or higher.
- GNU **G++** compiler 7.3.1 version or higher.
- **Make** tool for managing dependencies.
- File "**CMakeLists.txt**" in **DSPH-Chrono-Lib/Source**.
- **Project Chrono v4.0.0** already compiled.

Open CMake (cmake-gui). Paste the **DualSPHysics-Chrono/src_extra/DSPH-Chrono-Lib/Source** folder path into the textbox labelled as ***Where is the source code*** and paste the build folder path into the ***Where to build the binaries textbox***. Once the paths are introduced, the `Configure` button should be pressed. A new dialog will appear asking for the compiler to be used in the project. Please remember that **only GNU G++ compiler 7.3.1 version or higher for 64 bit are supported**.

Paste the **chrono-4.0.0/build/cmake** folder path in the textbox labelled as **Chrono_DIR**. This folder contains the properties of the Chrono project that was previously generated. The `Configure` button should be pressed again.

The output text should be ***Configuring done***. If the configuration was successful, now press the `Generate` button. This will generate a ***Makefile*** to compile the source files into the build directory and the output text should be ***Generating done***.

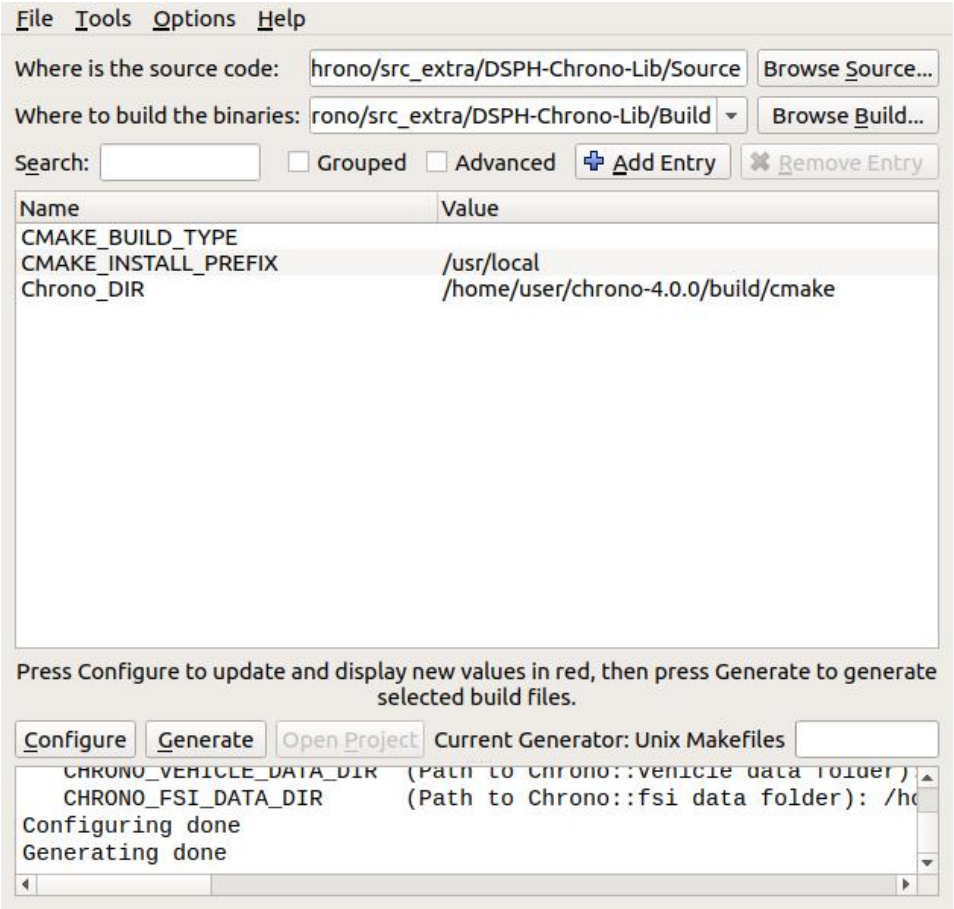


***Figure 3**. Building DSPHChronoLib with CMake*

Open a Linux terminal and go into **DualSPHysics-Chrono/src_extra/DSPH-Chrono-Lib/Build** folder and execute the command **make** on the terminal to compile the source code.

```
$ cd DSPH-Chrono-Lib/Build
$ make
```

If the compilation was successful, it will generate the file **libdsphchrono.so** into **DualSPHysics-Chrono/src_extra/DSPH-Chrono-Lib/Build**.
Paste this file into **DualSPHysics-Chrono/bin/linux** and into **DualSPHysics-Chrono/src/lib/linux_gcc**.

DualSPHysics

The source file **DualSphDef.h** contains the macro definitions to allow the coupling of DualSPHysics with **ChronoEngine**. Line **#define DISABLE_CHRONO** (in **DualSphDef.h**) should be commented out. In this package it is already commented out, so it is not needed to modify the file.

DualSPHysics provides a pre-generated ***Makefiles*** to compile the code. In **DualSPHysics-Chrono/src** there are also several folders:

- **lib/linux_gcc**: precompiled libraries for GCC.
- **source**: contains all the source files and *Makefiles*

The files ***libChronoEngine.so*** and ***libdsphchrono.so*** should be located in **DualSPHysics-Chrono/bin/linux** and in **DualSPHysics-Chrono/src/lib/linux_gcc** in order to compile with **Project Chrono**.

The ***Makefile*** includes the variable `DIRTOOLKIT` that must define the path where the CUDA toolkit directory is located. By default: `DIRTOOLKIT=/opt/cuda`.

Makefiles can be used to compile the code in linux:

`$ make –f Makefile`: full compilation just using make command

`$ make –f Makefile_cpu`: only for CPU

The result of the compilation is the executable ***DualSPHysics5.0_linux64*** or ***DualSPHysics5.0CPU_linux64*** created in **DualSPHysics-Chrono/bin/linux**.

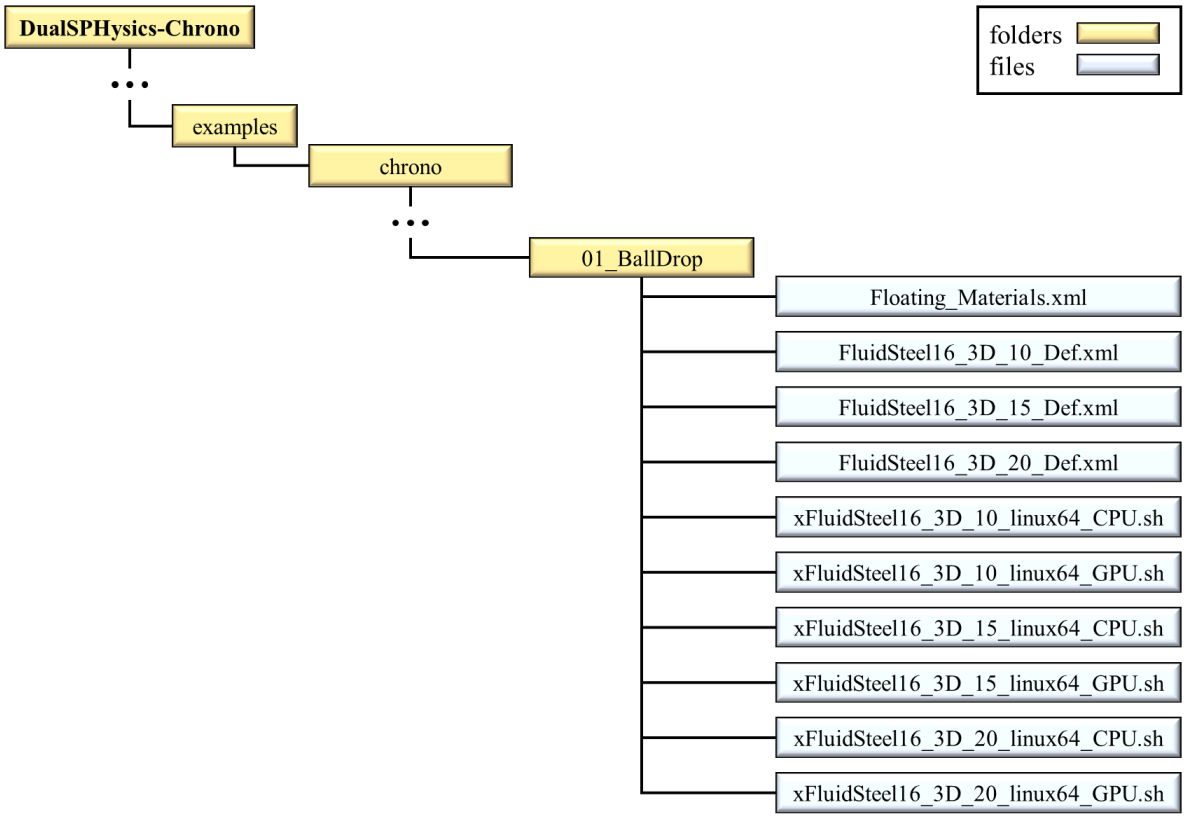
To exclude the use of OpenMP you have to remove the flags `–fopenmp` and `-lgomp` in the ***Makefile*** and comment line `#define OMP_USE` in **OmpDefs.h**. In order to simulate more than 2000 and up to 65000 floating or solid objects, the user should activate or comment the code line `#define CODE_SIZE4` in **Types.h**.

The user can modify the compilation options, the path of the CUDA toolkit directory, the GPU architecture. The GPU code is already compiled (bin) for compute capabilities sm30, sm35, sm50, sm52 sm61, sm70 and with CUDA v9.2.

Each ***Makefile*** includes a section with compilation options. Using YES/NO you can choose to compile or not **DualSPHysics** coupled with the external libraries.

Execution

This package contains 4 examples to execute the benchmarks proposed in the manuscript, located in **DualSPHysics-Chrono/examples/chrono**. The procedure to launch all the examples is the same, so this section is focused on the first benchmark (01_BallDrop).



The files included in the **01_BallDrop** folder are the following:

- **Floating_Materials.xml**: Contains the definition of the material properties.
- **Fluid16_3D_[10,15,20]_Def.xml**: Contains the initial configuration of the cases for the SPH resolutions dp=D/10, dp=D/15 and dp=D/20, where D=0.0167m is the diameter of the falling ball.
- **xFluidSteel16_3D_[10,15,20]_linux64_CPU.sh**: This script execute the case on CPU.
- **xFluidSteel16_3D_[10,15,20]_linux64_GPU.sh**: This script execute the case on GPU.

Give execution permissions

To prepare the environment the next steps should be followed in order to give the execution permissions for all the executables and scripts:

```
$ cd DualSPHysics-Chrono

$ chmod +x chpermissions.sh

$ ./chpermissions.sh
```

Run a case

To run the case **Fluid16_3D_10_Def.xml**, there are two scripts, **xFluidSteel16_3D_10_linux64_CPU.sh** and **xFluidSteel16_3D_10_linux64_GPU.sh**, for executions on CPU and GPU, respectively.

CPU mode:

```
$ cd examples/chrono/01_BallDrop

$ ./xFluidSteel16_3D_10_linux64_CPU.sh
```

GPU mode:

```
$ cd examples/chrono/01_BallDrop

$ ./xFluidSteel16_3D_10_linux64_GPU.sh
```

Both scripts (.sh) will execute the pre-processing tool **GenCase** to build the SPH domain. After, they will execute the **DualSPHysics** tool to run the simulation, which saves the information of the particles in binary files (.bi4) in the folder **Fluid16_3D_10_out/data**. Finally, several post-processing tools are used to analyse the simulation:

- **PartVTK_linux64**: Generates .VTK files to visualize the position of particles with the visualisation tool **ParaView**. These files also contain other magnitudes of the particles, like velocity and density.

- **FloatingInfo_linux64**: Generates .CSV files that contain the information of the required fluid-driven such as linear and angular velocities, displacement of the center of mass, motions and angles of rotation.