Lausanne, January 14, 2014

FlowKit Ltd
Route d'Oron 2
1010 Lausanne
Switzerland

Prof. Doug James
Cornell University
Ithaca, NY 14853
U.S.A

Free-Surface Palabos Starter Package

Free-Surface Model in Palabos

Palabos offers two different "volume-of-fluid" models for multi-phase flows, namely, the free-surface and the two-phase models. The essential difference between the two, is that the two-phase model solves for two fluids, but the free-surface model simulates only one fluid which evolves in an "empty" space of constant pressure. All codes provided for the Starter Package use the free-surface model which is described in the accompanying document `palabos_free_surface_model.pdf`.

Here, we shall only revisit the implemented models for "internal bubble pressure correction" with the free-surface model. In the free-surface model, air bubbles (in an air-water system) are not simulated, so they cannot conserve volume. Under gravity, these air bubbles simply collapse. In order to avoid this effect, Palabos uses an internal bubble pressure correction model. With this model the volume of the air bubbles is traced, and the pressure of the air bubbles is adjusted according to a relation of the form:

$$p = p(p_0, V_0, V),$$

where p is the new bubble pressure, p_0 is the reference bubble pressure, V_0 is the reference volume of the bubble, and V is the current volume of the bubble. The default model used in Palabos is:

$$p = p_0 \frac{V_0}{V},$$

so that if a bubble gets squeezed, its internal pressure will increase and it will resist collapsing. Palabos also has another model for the same purpose, defined as:

$$p = p_0 \left[1 + \alpha \left(1 - \frac{V}{V_0} \right)^{\beta} \right],$$

with:

$$\alpha \geq 0 \text{ and } \beta \geq 0.$$

The convention of Palabos is that if the user provides a value of α or β which is less than zero, the first bubble pressure model is used.

Falling Jet in a Pool of Water

The first code of the Starter Package is called `fallingJet`. It is used to simulate a water jet emerging from an orifice and falling in a pool of water. The files provided are the source code `fallingJet.cpp`, the input configuration file `fallingJet.xml` and the `Makefile`.

To compile the code, the user only needs to edit the `Makefile`, adjust the parameters to his own needs (compiler name, compilation for parallel execution, etc) and just type `make` at the working directory. For parallel execution a version of the MPI library must also be installed in the user's system. To execute the program in parallel with, say 32 processes, one needs to use:

```
mpirun -np 32 ./fallingJet fallingJet.xml
```

The input configuration file `fallingJet.xml` must be given as a command-line parameter. The code produces terminal output when it is executed, and it also produces several files for post-processing which are placed in the `./tmp` directory.

The simulation is fully configured from the `fallingJet.xml` file, so the user needs to compile the application only once. The input XML file can be adjusted to the user's needs. In the provided file there exist a lot of comments, so the meaning of all parameters should be more or less clear. The physical properties of the fluid material are defined from three dimensionless quantities, namely, the Reynolds, Weber and Bond numbers. The contact angle given in the XML file is in degrees, and if it is less than zero, then the contact angle algorithm is deactivated. The user has to provide an inlet velocity U of the water at the orifice. He also has to provide a lattice velocity U^{lb}, which is much lower than 1.0 and is used to compute the time step from the following relation:

$$\delta_t = \frac{U^{lb}}{U}\delta_x,$$

with the spatial step being:

$$\delta_x = \frac{R}{N-1},$$

where R is the radius of the orifice, and N is the `resolution` parameter in the XML input file. If a bubble pressure model is used, then the parameter `bubbleVolumeRatio` is of great importance. In the proceeding section we saw that the pressure of the air bubbles depends on a reference volume V_0. This volume is computed by multiplying the `bubbleVolumeRatio` parameter, with the actual computed volume of the air bubble at the moment of its creation. `bubbleVolumeRatio` must be greater equal to 1.0, and the bigger it is, the stronger the air bubbles oppose to their collapsing.

As mentioned above, all output files for post-processing are saved in the local `./tmp` directory. There, the user can find three kinds of files: `log` files, `stl` files and `vti` files. The `log` files are text files which contain information about the bubble creation, evolution and properties, when the internal bubble pressure model is activated. The `bubbleTimeHistory.log` and `fullBubbleRecord.log` files contain values in lattice units. This means that length values should be multiplied by δ_x and volume values by δ_x^3 to go to physical units. The file `bubbles.log` contains everything in physical units. A word of caution is necessary here. When logging and displaying results, we consider the bubbles to be spherical, so they have a center and a radius. This is approximate, and the user is warned that sometimes the bubbles have other shapes (like toroidal shapes for instance), so a "bubble radius" cannot be defined. All the rest files to be described, contain in their file-name the respective iteration during which they were written to disk. The `stl` files contain triangular surface meshes of the air-water interface. The interface is defined as the iso-surface for the value 0.5 of the volume fraction (the volume fraction is the volume ratio of water in the volume of a computational cell). Sometimes we apply a Laplacian filter to the volume fraction field before computing the iso-surface or exporting the volume fraction for better visualization. In such cases, we refer to the respective quantities as "smoothed". There are also `stl` files that contain the bubbles, as represented by spheres (see the comment above about this spherical approximation). A comment on terminology is is order here. In the log files, we refer to some bubbles as being "frozen". This means that the internal bubble pressure correction algorithm does not apply for them. The ambient space is always treated as a "frozen" bubble. Frozen bubbles are not logged in some files, and they are not represented as spheres in the respective `stl` files. The last kind of output files is the

"volume data" `vti` files. These files are to be post-processed with the `Paraview` software. They contain information such as bubble tags and volume fraction for the whole simulation domain.

Rising Bubble Inside a Pool of Fluid

This code was not asked by Cornell, but we include it in the Starter Package in order to demonstrate the capabilities and the limitations of the implemented bubble pressure correction model in Palabos. All the rest of the codes in the Starter Package are similar to the already described `fallingJet` code in terms of compilation and usage, so from now on, we shall describe only the important parameters that need to be set by the user in the corresponding input XML files.

The `risingBubble` test case describes an air bubble rising in a pool of fluid. The bubble has an initial spherical shape. The setting of the problem is non-dimensionalized, so that the full computational box has an edge of length 1.0. The time scale of the problem is set by the gravitational acceleration, so the value of the acceleration of gravity is 1.0 in dimensionless units. The physical parameters of the simulation are determined by two dimensionless numbers. The Jesus number `Je` is used to determine the value of surface tension in the simulation. If the Jesus number is 0, then surface tension is not modeled. The Galilei number `Ga` is used to define the kinematic viscosity of the fluid. Since the time scale is set by gravity and the gravitational acceleration cannot be 0, the Galilei number can never be 0 either. All the other parameters included in the input XML file are similar to the ones in the `fallingJet` code and can be understood by reading the comments inside the file.

This code, when executed even in very low resolution as the default values in the XML file determine, demonstrates that the usage of the bubble pressure correction model can support the existence and the volume of air bubbles inside the fluid. The simulated bubble conserves its volume and rises because of the pressure difference inside the fluid pool. If the bubble pressure correction algorithm is deactivated (by commenting out the relevant lines of code in the source file) then the air bubble progressively vanishes and we end up with a totally unphysical situation. Under the `risingBubble` directory you can find a directory `animations` which contains two movies of the two simulations just described. Please keep in mind that these simulations were conducted only to provide a "proof of concept" for the bubble pressure correction model and not to study the detailed physics of rising bubbles. For something like that, the resolution of the simulations should be considerably higher.

Falling Droplet in a Pool of Fluid

The next code of the Starter Package simulates the motion of a spherical droplet falling from a certain height inside a pool of fluid. Gravity acts in the -z direction. The problem is again non-dimensionalized, with a reference length equal to the edge of the computational box in the x-direction (edge length of the fluid pool in the x-direction). The fluid pool height, the position and radius of the droplet can be defined conveniently by the user.

There are three important simulation parameters, namely the initial velocity of the droplet, the acceleration of gravity and the kinematic viscosity of the fluid. There are three dimensionless numbers that define these three simulation parameters, the Reynolds, Galilei and Laplace numbers. The Reynolds number is used to define the initial velocity of the droplet. If it is less or equal to 0 then the droplet has no initial velocity. The Galilei number is utilized to determine the acceleration of gravity. If it has a value less or equal to 0, then gravity is not included in the numerical model. Finally, the Laplace number is used to set the surface tension. It can acquire a value of 0 in which case surface tension is not modeled. More information on how to use these three dimensionless numbers can be found in the comments of the corresponding input XML file. After the user sets

his desired spatial resolution (by the `resolution` parameter in the XML file), the time step is determined by the value `uLB` of the lattice velocity, which is defined as the ratio of δ_t over δ_x. The rest of the user-configured simulation parameters are similar to the other codes in the Starter Package.

Again, there exists a directory called `animations` which contains the results of a low resolution simulation of the falling droplet code with the default parameters in the input configuration file. The user can change all physical parameters according to his convenience and the simulation will lead to a more or less "splashy" result.

Dam Break with an Solid Obstacle

This code of the Starter Package simulates a classical dam break problem with the addition that there also exists a solid obstacle in the path of the moving fluid, on which the fluid impacts and splashes. Gravity again acts in the -z direction, and the problem is non-dimensionalized by a reference length equal to the height of the full computational box. The geometry of this problem cannot be changed from the configuration XML file. To change it one must edit the source code `damBreak.cpp`. The material properties of the fluid can be handled by providing the two dimensionless numbers `Je` and `Ga` which are previously described in the rising bubble section. The code and the input XML file are relatively straightforward. The animated results of a medium resolution simulation are also included in the provided files.