Project work for the PhD course in OpenFOAM

Roll Motion of a Box and Interaction with Free-Surface

Arash Eslamdoost

arash.eslamdoost@chalmers.se

Computational Hydrodynamics
Sustainable Ship Propulsion
Shipping and Marin Technology Department
Chalmers University of Technology
Gothenburg, Sweden

December 2009

1. Introduction

The objective of this tutorial is to investigate the capability of OpenFOAM-1.6.x in handling free-surface flows when there is some mesh deformation in computational domain. Such a problem could be seen a lot in ship motion modeling. To simplify the problem enough, initially it is considered that there is a two-dimensional box located between the interface of water and air. Moreover, to check the OpenFOAM's capability in handling dynamic mesh, the box will be set to roll in an oscillating mode which requires adapting the grid for each time step. The general presentation of the discussed modeling could be seen in Figure 1.

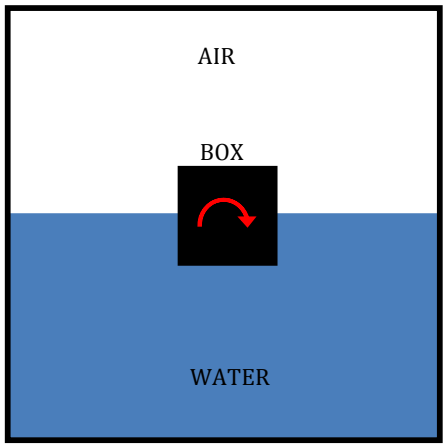

Figure 1. Schematic presentation of the intended modeling

It is common among OpenFOAM users to brows and find the available modeling in OpenFOAM tutorial which are more similar to the new intended modeling and base this new modeling on these previously linked solvers and libraries. The most similar available tutorial to the rolling box case is "sloshingTank2D", which could be found in the following directory:

$FOAM_TUTORIALS/multiphase/interDyMFoam/ras/sloshingTank2D

In "sloshingTank2D" tutorial there is a tank with partially filled with water. It is possible to give some roll and other sort of motions (e.g. sway and heave) to the whole tank together. Therefore, in this case, there is a moving mesh but actually there is not any change in cell volumes and arrays relative to each other. This case is running with the "interDyMFoam" solver and according to the information provided in [1] this solver is applied for two incompressible, isothermal immiscible fluids using a VOF (volume of fluid) phase-fraction based interface capturing approach, with optional mesh motion and mesh topology changes including adaptive re-meshing. Therefore, "interDyMFoam" solver should be able to handle both interface tracking and mesh motion in the rolling box case.

As a first step to start the rolling box modeling, it has been tried to just change the geometry of the tank and replace it with the rolling box geometry. Now, it is required to generate a new grid and this is done through the blockMesh application. The detailed coordinate of the vertexes, edges, patches, blocks and mesh adaption implemented in the blockMesh file is presented in appendix A. The generated mesh is

rather coarse and in case that the accuracy of the solution is important this grid should be refined.

Now on, the case which the whole computational domain is supposed to have an angular oscillation about a defined axis, is called “boxSolidOscillation” and the case, which only the box is supposed to oscillate, is called “boxFreeOscillation”. These cases could be found in the OpenFOAM Course Homepage in the following address:

http://www.tfd.chalmers.se/~hani/kurser/OS_CFD_2009/

Download and copy them to your run directory and then unpack the contents.

2. boxSolidOscillation case

Current case just deals with pure solid movement of whole computational domain without having any change in cells’ volume. The dynamic mesh handling and controlling parameters on moving the whole field in this case will be discussed further in the following.

To get started with this case, move in to the “boxSolidOscillation” directory.

```
cd boxSolidOscillation
```

Created “boxSolidOscillation” folder as usual contains three sub-directories, which are “0”, “constant”, and “system”. Some of the main contents of these sub-directories will be discussed a little more in deep.

Grid applied in this case is produced applying blockMesh. To check the grid properties open “constant/polyMesh/blockMesh”. Also, you might find the blockMesh file contents in appendix A. Run the blockMesh to generate the mentioned structured grid and then to check the mesh quality the “checkMesh” command could be applied.

```
blockMesh
checkMesh
```

Open and investigate the generated grid in paraview. It can bee seen that the generated mesh is located on xy-plane. Default dynamicMesh is set in a way that the axis of angular oscillation should be located along x-axis; but, current generated grid is located on xy-plane and its axis of oscillation is along z-axis. According to this, the grid should be adapted by changing the current array of x and z axis. Issuing the following command line could do this:

```
rotateMesh “(0 0 1)” “(1 0 0)”
```

Now it is required to modify the files in “0” directory for current geometry and set the proper boundary conditions. Appendixes 2, 3, 4 present the settings for velocity, pressure and alpha1 boundary condition. Boundaries of box and tank are treated as non-moving walls. Pressure boundary conditions on walls are treated as

"bouyantPressure". Also, "zeroGradient" alpha1[1] has been set on boundaries.

To see how the initial phases distribution in the computational domain is defined open "system/setFields". Settings for "setFields" application could be seen in Appendix 5.

In the fvSolution, the PISO sub-dictionary contains elements that are specific to interFoam. There are the usual correctors to the momentum equation but also correctors to a PISO loop around the α_1 phase equation. Of particular interest are the nAlphaSubCycles and cAlpha keywords. nAlphaSubCycles represents the number of sub-cycles within the α_1 equation; sub-cycles are additional solutions to an equation within a given time step. It is used to enable the solution to be stable without reducing the time step and vastly increasing the solution time. Here 3 sub-cycles has been specified, which means that the α_1 equation is solved in 3× one-third-length time steps within each actual time step [2].

The cAlpha keyword is a factor that controls the compression of the interface where: 0 corresponds to no compression; 1 corresponds to conservative compression; and, anything larger than 1, relates to enhanced compression of the interface [2].

```
PISO
{
    momentumPredictor           no;
    nCorrectors                 2;
    nNonOrthogonalCorrectors    0;
    nAlphaCorr                  1;
    nAlphaSubCycles             3;
    cAlpha                      1.5;
    correctPhi                  no;

    pRefPoint                   (0 -4 0);
    pRefValue                   1e5;
}
```

Phases (water and air) initial distribution is fixed by applying the "setFields" command.

```
Cp 0/alpha1.org 0/alpha1
SetFields
```

Mesh movement in this case is applied through the available dictionary in "constant" directory, which is called "dynamicMeshDict".

```
dynamicFvMesh   solidBodyMotionFvMesh;

solidBodyMotionFvMeshCoeffs
{
    solidBodyMotionFunction SDA;
    SDACoeffs
    {
```

[1] alpha1 is the phase fraction of phase 1 in a multi-phase system, alpha in the control dictionaries refers to any phase. The reason for the "inconsistency" is to ensure that the discretisation schemes are the same for all phases.

```
    CofG          ( 0 0 0 );
    lamda         50;
    rollAmax      0.22654;
    rollAmin      0.10472;
    heaveA        0;
    swayA         0;
    Q             2;
    Tp            13.93;
    Tpn           11.93;
    dTi           0.059;
    dTp          -0.001;
  }
}
```

Ship design analysis (SDA) mentioned in the "dynamicMeshDict" is a 3DoF motion function, which is applied to the whole computational cells all together. SDA class comprises sinusoidal roll (rotation about x), heave (z-translation) and sway (y-translation) motions with changing amplitude and phase. It could be found in the following directory:

$FOAM_SRC/dynamicFvMesh/solidBodyMotionFvMesh/solidBodyMotionFunctions

In "dynamicMeshDict" there are some coefficients required for defining the way that the computational domain is supposed to move. These coefficients are introduced in "SDA.H" as private data and going to be employed in motion equations presented in "SDA.C". Definition and dimension for each of these coefficients are presented in Table 1.

Table 1. Coefficients applied in "dynamicMeshDict"

Coefficient	Description	Type	Dimension
CofG	Center of gravity	vector	[m]
lamda	Model scale ratio	scalar	[-]
rollAmax	Max roll amplitude	scalar	[rad]
rollAmin	Min roll amplitude	scalar	[rad]
heaveA	Heave amplitude	scalar	[m]
swayA	Sway amplitude	scalar	[m]
Q	Damping Coefficient	scalar	[-]
Tp	Time Period for liquid	scalar	[sec]
Tpn	Natural Period of Ship	scalar	[sec]
dTi	Reference time step	scalar	[sec]
dTp	Increase in Tp per unit dTi	scalar	[-]

In current case just the roll motion of the computational domain is of our interest and therefore the heave and sway coefficients are set to zero.

Now, the case is ready to run applying "interDyMFoam" application.

```
interDyMFoam
```

This application will start the solution first by reading the scheme of mesh motion and then setting the initial fields such as g, p, alpha1, U and boundary conditions. Consequently, the incompressible transport model and turbulence model are set. Presented lines in the following gray box is the report of this procedure that appears

on screen after issuing the interDyMFoam command. Moreover, output of running this application is presented for one of the time steps. According to setting the nAlphaSubCycles to 3, it could be seen that three consequent iterations occurs for solving the multidimensional universal limiter for explicit solution (MULES).

```
Create time
Create mesh for time = 0

Selecting dynamicFvMesh solidBodyMotionFvMesh
Selecting solid-body motion function SDA

Reading g
Reading field p

Reading field alpha1

Reading field U

Reading/calculating face flux field phi

Reading transportProperties

Selecting incompressible transport model Newtonian
Selecting turbulence model type laminar
time step continuity errors : sum local = 0, global = 0, cumulative = 0
GAMGPCG:  Solving for pcorr, Initial residual = 1, Final residual = 2.2321e-13, No Iterations 1
time step continuity errors : sum local = 1.38355e-11, global = 5.61669e-30, cumulative = 5.61669e-30
Courant Number mean: 1.3569e-11 max: 1.72618e-10

Starting time loop
.
Courant Number mean: 0.0930747 max: 0.472539
deltaT = 0.00714286
Time = 0.128571

solidBodyMotionFunctions::SDA::transformation(): Time = 0.128571 transformation: ((0 0 0) (0.999235 (0.0391004 0 0)))
Execution time for mesh.update() = 0.01 s
MULES: Solving for alpha1
Liquid phase volume fraction = 0.5  Min(alpha1) = -4.02279e-20  Max(alpha1) = 1
MULES: Solving for alpha1
Liquid phase volume fraction = 0.5  Min(alpha1) = -2.78346e-21  Max(alpha1) = 1
MULES: Solving for alpha1
Liquid phase volume fraction = 0.5  Min(alpha1) = -6.47083e-22  Max(alpha1) = 1
GAMG:  Solving for p, Initial residual = 0.0681279, Final residual = 0.000220229, No Iterations 2
time step continuity errors : sum local = 1.0173e-06, global = 1.50675e-17, cumulative = -1.83543e-16
GAMGPCG:  Solving for p, Initial residual = 0.00106785, Final residual = 1.71041e-09, No Iterations 7
time step continuity errors : sum local = 7.4766e-12, global = 1.50911e-17, cumulative = -1.68452e-16
ExecutionTime = 1.18 s  ClockTime = 1 s
```

Result of the computation is presented in two sets of phase distribution and velocity field in Figure 2.

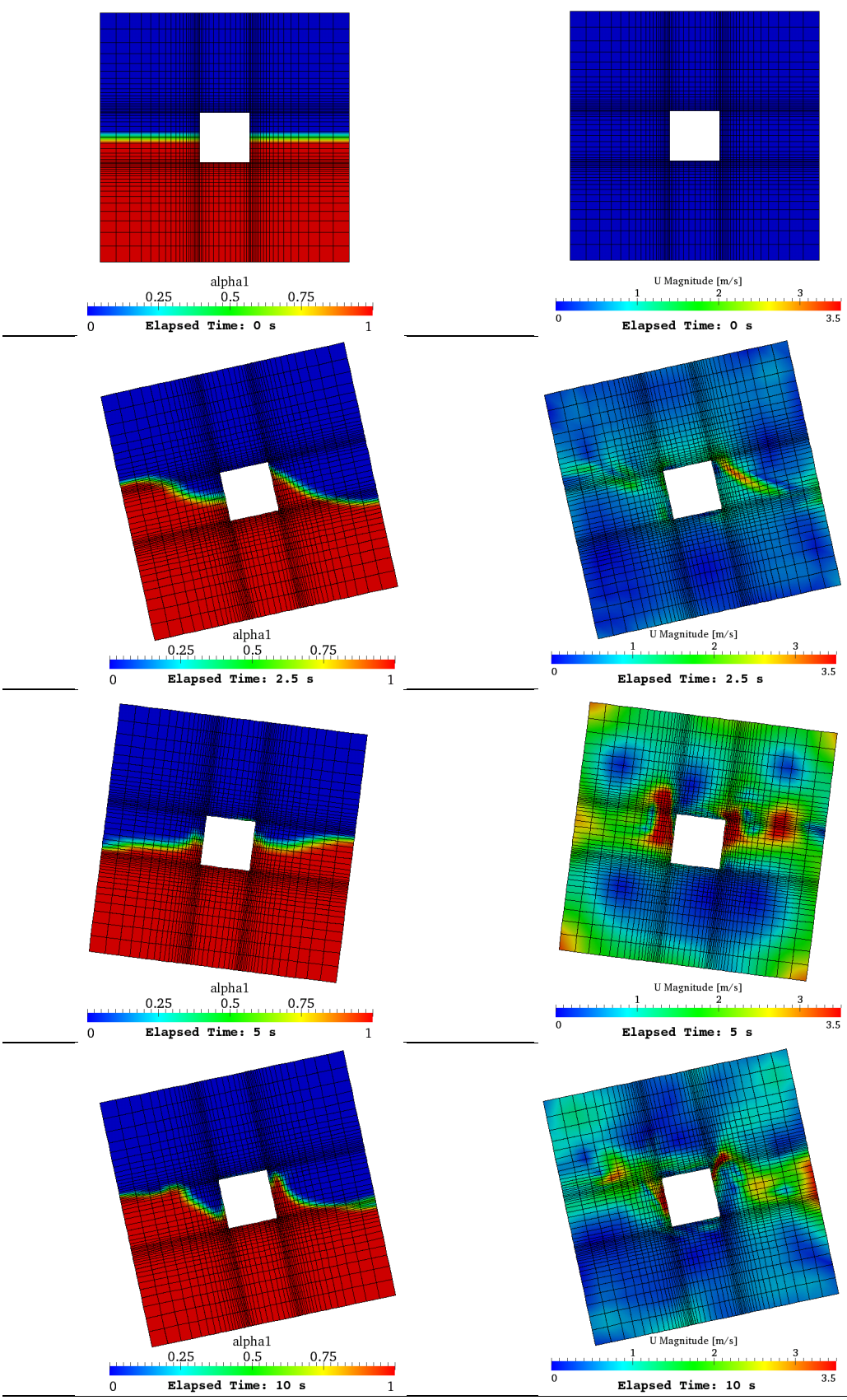

Figure 2. **Column A presents the alpha1 distribution and column B shows the velocity field for the corresponding time**

3. boxFreeOscillation

In this section, in contrast with the previous one, instead of moving the whole computational grid as a solid body just the inner box is going to have an angular oscillation. So, it is required to set up another application for the dynamic mesh treatment. At this stage, difference between boxFreeOscillation case and boxSolidOscillation is between the settings of "dynamicMeshDict" located at "constent" directory and consequently the required boundary motion input data in "pointMotionU" located in "0" directory. Moreover, it is required to modify the "fvSolution" dictionary and define the cell motion solver. These modification and settings are going to be discussed in the following sections.

3.1. meshMotionLib

At the Moment, there are two available libraries in OpenFoam-1.6.x, which can produce the desired oscillating for specified patches. They are called "angularOscillatingVelocity" and "angularOscillatingDisplacement" and both could be found in the following path:

$FOAM_SRC/fvMotionSolver/pointPatchFields/derived/

Both of these libraries are applicable for current case based on user demand. Here, "angularOscillatingVelocity" library is going to be used. This library is attached to the materials available on OpenFOAM Course homepage in "meshMotionLib" directory. To be able to make some modification in this library and probably prepare the library to a specific type of motion it would be better to edit the "files" and "option" files and recompile it to the user library. A general description to "angularOscillatingVelocity" is presented in the following paragraphs.

3.1.1. angularOscillatingVelocity

The source files for this library could be found in "dynamicMeshDict" folder of the provided materials at OpenFOAM Course homepage.

```
cd dynamicMeshDict
```

The original ".H" and ".C" files copied from the "$FOAM_SRC/fvMotionSolver/pointPatchFields/derived/angularOscillatingVelocity" path were named "libOscillatingVelocityPointPatchVectorField.H" and "libOscillatingVelocityPointPatchVectorField.C" but in order not to make some distinction among the original library and this one the string "libOscillatingVelocity" in the names of ".H" and ".C" files has been replaced with "libMyOscillatingVelocity". This replacement has to be done inside these source files as well as "files" and "options" in the "Make" directory. Finally, the "files" and "options" files should be the same as following ones.

files

```
libMyOscillatingVelocityPointPatchVectorField.C

LIB = $(FOAM_USER_LIBBIN)/ libMyOscillatingVelocityPointPatchVectorField
```

options

```
EXE_INC = \
    -I$(LIB_SRC)/triSurface/lnInclude \
    -I$(LIB_SRC)/meshTools/lnInclude \
    -I$(LIB_SRC)/dynamicMesh/lnInclude \
    -I$(LIB_SRC)/finiteVolume/lnInclude \
    -I$(LIB_SRC)/fvMotionSolver/lnInclude

LIB_LIBS = \
    -ltriSurface \
    -lmeshTools \
    -ldynamicMesh \
    -lfiniteVolume
```

The main part of this library which defines the movement of the patches is presented as member function in "libOscillatingVelocityPointPatchVectorField.C". Here, the velocity of each point on a specific patch is calculated for each time step. One may define a new function to move the patches by editing the presented function in this file and if required the constructors in ".H" file. The mechanism of oscillating velocity function could be seen in 0. In a specific case that the origin of rotation is located in the center of the box the result of the motion should be a pure roll around box's center axis. The function of this oscillation for patches, which exists as a member functions, is presented in the following gray box.

libOscillatingVelocityPointPatchVectorField.C

```
// * * * * * * * * * * * * * * * * Member Functions  * * * * * * * * * * * * * * //

void angularOscillatingVelocityPointPatchVectorField::updateCoeffs()
{
    if (this->updated())
    {
        return;
    }

    const polyMesh& mesh = this->dimensionedInternalField().mesh()();
    const Time& t = mesh.time();
    const pointPatch& p = this->patch();

    scalar angle = angle0_ + amplitude_*sin(omega_*t.value());
    vector axisHat = axis_/mag(axis_);
    vectorField p0Rel = p0_ - origin_;

    vectorField::operator=
    (
        (
            p0_
          + p0Rel*(cos(angle) - 1)
          + (axisHat ^ p0Rel*sin(angle))
          + (axisHat & p0Rel)*(1 - cos(angle))*axisHat
          - p.localPoints()
        )/t.deltaT().value()
    );

    fixedValuePointPatchField<vector>::updateCoeffs();
}
```

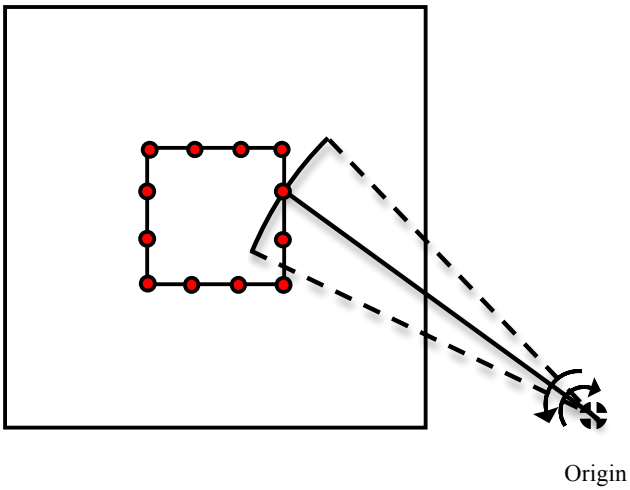

Figure 3. Mechanism of oscillating velocity

Now, to make current library available for the other applications, it should be compiled through following command.

```
wmake libso
```

There should be a link to this new library in "controlDict" file to let the employed applications to know about it. Inserting the following line to the "controlDict" does this.

```
libs ("libMyOscillatingVelocityPointPatchVectorField.so");
```

After preparing the meshMotion library, it is required to do some settings in the "fvSoloution" dictionary and define the solver for mesh motion application. This purpose is done by adding the following lines to the solvers in the "fvSolution".

fvSolution

```
cellMotionUx PCG
  {
    preconditioner   DIC;
    tolerance      1e-08;
    relTol         0;
  };

  cellMotionU PCG
  {
    preconditioner   DIC;
    tolerance      1e-08;
    relTol         0;
  };
```

"solvers" specifies each linear-solver that is used for each discretised equation. The syntax for each entry within "solvers" uses a keyword that is the word relating to the variable being solved in the particular equation. The choises for "solvers" are presented in Table 2 [1].

Table 2. Solver Options

Solver	Keyword
Preconditioned (bi-)conjugate gradient	PCG/PBiCG*
Solver using a smoother	smoothSolver
Generalised geometric-algebraic multi-grid	GAMG

*PCG for symmetric matrices, PBiCG for asymmetric

There is a range of options for preconditioning of matrices in the conjugate gradient solvers, represented by the preconditioner keyword in the solver dictionary. The preconditioners are listed in Table 3 [1].

Table 3. Preconditioner options

Preconditioner	Keyword
Diagonal incomplete-Cholesky (symmetric)	DIC
Faster diagonal incomplete-Cholesky (DIC with caching)	FDIC
Diagonal incomplete-LU (asymmetric)	DILU
Diagonal	diagonal
Geometric-algebraic multi-grid	GAMG
No preconditioning	none

The other distinction in "boxFreeOscillating" case comparing to the "boxSolidOscillating" case is in "dynamicMeshDic" dictionary located in "constant" directory. The new dictionary is presented in the following gray box.

dynamicMeshDict

```
FoamFile
{
   version    2.0;
   format     ascii;
   class      dictionary;
   location   "constant";
   object     motionProperties;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dynamicFvMesh   dynamicMotionSolverFvMesh;
motionSolverLibs ("libfvMotionSolvers.so");
solver velocityLaplacian;

diffusivity uniform;
```

It could be seen that there are two parameters that are important in dynamic mesh manipulation which are "solver" and "diffusivity" scheme. Available models for the "solver" are:

- displacementLaplacian
- velocityLaplacian
- SBRStress

which in "boxFreeOscillating" case has set to be velocityLaplacian. Available options for diffusivity models are presened in Table 4. Further information on "solver" and "diffusivity" models could be found in [5].

Table 4. Diffusivity Models

quality-based methods	distance-based methods*
• uniform	• linear
• directional	• quadratic
• motionDirectional	• exponential
• inverseDistance	

*These models are used with "inverseDistance" method

Now it is required to define the motion of the moving patches (box) in "pointMotionU" file located in "0" directory. This input file is required to calculate the coordinate of each point on a specified moving patch during time variation. The motion equation was reviewed previously. (libOscillatingVelocityPointPatchVectorField.C)

Available basic parameters in "pointMotionU" which control the motion of moving patches are presented in Table 5.

Table 5. Motion control Parameters in "pointMotionU"

Parameter	Type	Description	Dimension
axis	Vector	Axis of Rotation	m
origin	Vector	Center of Rotation	m
angle	Scalar	Oscillation Occurs this reference angle	rad
amplitude	Scalar	Amplitude of Oscillations	m
omega	Scalar	Angular Frequency	rad/sec

pointMotionU

```
FoamFile
{
   version 2.0;
   format ascii;
   class pointVectorField;
   object pointMotionU;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
dimensions     [0 1 -1 0 0 0 0];
internalField   uniform (0 0 0);
boundaryField
{
  tank
   {
      type          fixedValue;
      value         uniform (0 0 0);
   }
   box
   {
      type ibMyOscillatingVelocityPointPatchVectorField;
      axis (1 0 0);
      origin (0 100 0);
      angle0 0;
      amplitude 0.01;
      omega 5;
      value uniform (0 0 0);
   }
   front
   {
      type          empty;
   }
   back
   {
      type          empty;
   }
}
```

After setting a dynamic mesh solver for the case, it is the time for setting the phase (alpha) distribution in the computational domain. This step is totally similar to the "setFields" in previous case. Therefore, "setFieldsDict" dictionary settings should be similar to the appendix 4 then it is possible to run the following command:

```
Cp 0/alpha1.org 0/alpha1
setFields
```

At the moment needed setting are done and the case is ready to start the calculations applying interDyMFoam.

```
interDyMFoam
```

Results of the "boxFreeOscillating" case with presented settings are depicted in Figure 4. Paying attention to the phase distribution graphs one can see that the maximum amount of alpha1 is greater than 1, which is not practical and should be identical to 1. This problem appears to be originated from the MULES implicit solver in the interPhaseChangeFoam and should be fixed.

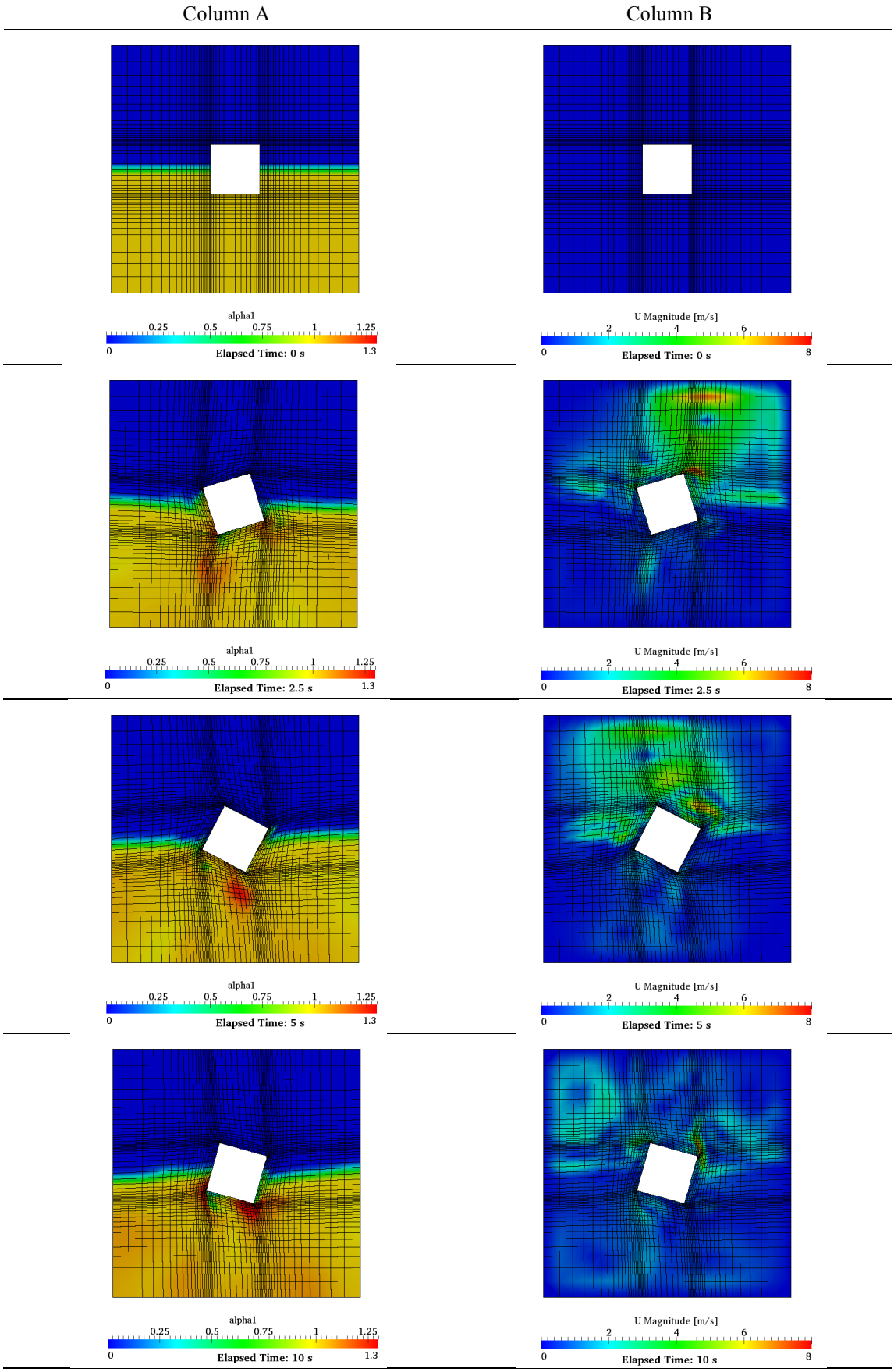

Figure 4. Column A presents the alpha1 distribution and column B shows the velocity field for the corresponding time

References:

1. *http://www.cfd-online.com/Forums/openfoam/*

2. *http://www.opencfd.co.uk/openfoam/doc/damBreak.html*

3. Eysteinn Helgason, *Point-wise deformation of mesh patches*, Project work for the PhD course in OpenFOAM, Chalmers University of Technology, Göteborg, Sweden, Spring 2009.

4. Erik Ekedahl, *6-DOF VOF-solver without Damping in OpenFOAM*, Project work for the PhD course in OpenFOAM, Chalmers University of Technology, Göteborg, Sweden, Winter 2008.

5. Pirooz Moradnia, *A tutorial on how to use Dynamic Mesh solver IcoDyMFOAM*, Project work for the PhD course in OpenFOAM, Chalmers University of Technology, Göteborg, Sweden, Spring 2008.

Appendix A
Grid Generation Applying blockMesh

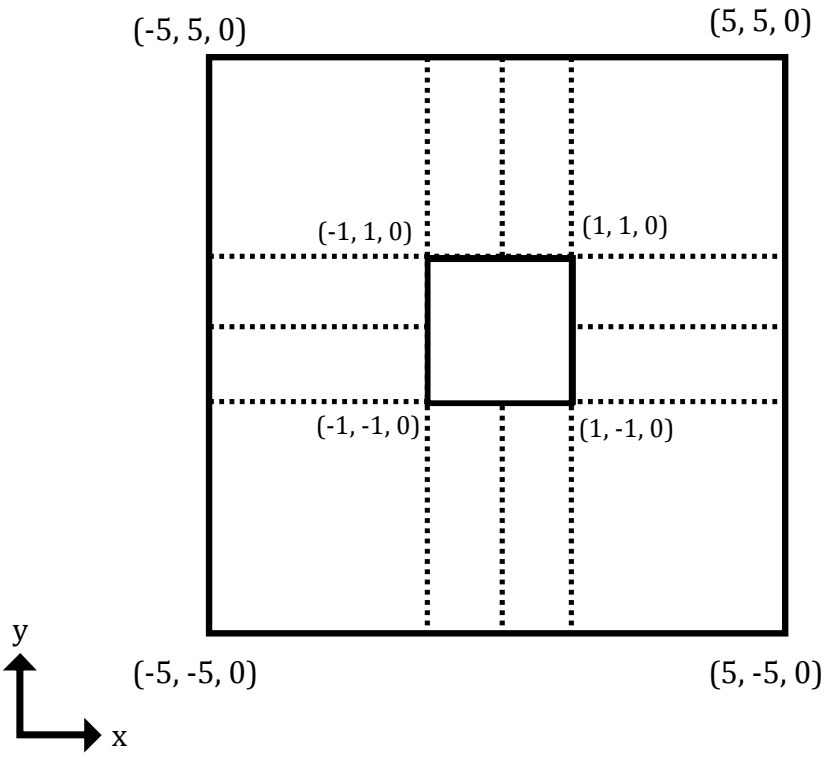

Figure A. 1: Dimensions of the generated grid

Following Lines presents the grids generated and employed in this tutorial.

```
// Parametric description
vertices
(
// Back Vertexes /////////
   (-5 -5 -0.5)   //Vertex No. = 0
   (-1 -5 -0.5)   //Vertex No. = 1
   (0 -5 -0.5)    //Vertex No. = 2
   (1 -5 -0.5)    //Vertex No. = 3
   (5 -5 -0.5)    //Vertex No. = 4

   (-5 -1 -0.5)   //Vertex No. = 5
   (-1 -1 -0.5)   //Vertex No. = 6
   (0 -1 -0.5)    //Vertex No. = 7
   (1 -1 -0.5)    //Vertex No. = 8
   (5 -1 -0.5)    //Vertex No. = 9
```

```
    (-5 0 -0.5)    //Vertex No. = 10
    (-1 0 -0.5)    //Vertex No. = 11
    (1 0 -0.5)     //Vertex No. = 12
    (5 0 -0.5)     //Vertex No. = 13

    (-5 1 -0.5)    //Vertex No. = 14
    (-1 1 -0.5)    //Vertex No. = 15
    (0 1 -0.5)     //Vertex No. = 16
    (1 1 -0.5)     //Vertex No. = 17
    (5 1 -0.5)     //Vertex No. = 18

    (-5 5 -0.5)    //Vertex No. = 19
    (-1 5 -0.5)    //Vertex No. = 20
    (0 5 -0.5)     //Vertex No. = 21
    (1 5 -0.5)     //Vertex No. = 22
    (5 5 -0.5)     //Vertex No. = 23

//Front Vertexes ///////
    (-5 -5 0.5)    //Vertex No. = 24
    (-1 -5 0.5)    //Vertex No. = 25
    (0 -5 0.5)     //Vertex No. = 26
    (1 -5 0.5)     //Vertex No. = 27
    (5 -5 0.5)     //Vertex No. = 28

    (-5 -1 0.5)    //Vertex No. = 29
    (-1 -1 0.5)    //Vertex No. = 30
    (0 -1 0.5)     //Vertex No. = 31
    (1 -1 0.5)     //Vertex No. = 32
    (5 -1 0.5)     //Vertex No. = 33

    (-5 0 0.5)     //Vertex No. = 34
    (-1 0 0.5)     //Vertex No. = 35
    (1 0 0.5)      //Vertex No. = 36
    (5 0 0.5)      //Vertex No. = 37

    (-5 1 0.5)     //Vertex No. = 38
    (-1 1 0.5)     //Vertex No. = 39
    (0 1 0.5)      //Vertex No. = 40
    (1 1 0.5)      //Vertex No. = 41
    (5 1 0.5)      //Vertex No. = 42

    (-5 5 0.5)     //Vertex No. = 43
    (-1 5 0.5)     //Vertex No. = 44
    (0 5 0.5)      //Vertex No. = 45
    (1 5 0.5)      //Vertex No. = 46
    (5 5 0.5)      //Vertex No. = 47
);

//Blocks//////
blocks
(
        hex (0 1 6 5 24 25 30 29)    (20 20 1)  simpleGrading  (0.2 0.2 1)   //Block No. = 0
        hex (1 2 7 6 25 26 31 30)    (10 20 1)  simpleGrading  (3 0.2 1)       //Block No. = 1
        hex (2 3 8 7 26 27 32 31)    (10 20 1)  simpleGrading  (0.3333 0.2 1)   //Block No. = 2
        hex (3 4 9 8 27 28 33 32)    (20 20 1)  simpleGrading  (5 0.2 1)       //Block No. = 3
        hex (5 6 11 10 29 30 35 34)   (20 10 1)  simpleGrading  (0.2 3 1)   //Block No. = 4
```

```
        hex (8 9 13 12 32 33 37 36)   (20 10 1)  simpleGrading  (5 3 1)       //Block No. = 5
        hex (10 11 15 14 34 35 39 38) (20 10 1)  simpleGrading  (0.2 0.3333 1)   //Block No. = 6
        hex (12 13 18 17 36 37 42 41) (20 10 1)  simpleGrading  (5 0.3333 1)       //Block No. = 7
        hex (14 15 20 19 38 39 44 43) (20 20 1)  simpleGrading  (0.2 5 1)   //Block No. = 8
        hex (15 16 21 20 39 40 45 44) (10 20 1)  simpleGrading  (3 5 1)       //Block No. = 9
        hex (16 17 22 21 40 41 46 45) (10 20 1)  simpleGrading  (0.3333 5 1)   //Block No. = 10
        hex (17 18 23 22 41 42 47 46) (20 20 1)  simpleGrading  (5 5 1)       //Block No. = 11
);
//Edges////////
edges
(
);
//Patches///////
patches
(
   wall box
   (
      (6 30 35 11)
      (11 35 39 15)
      (15 39 40 16)
      (16 40 41 17)
      (17 41 36 12)
      (12 36 32 8)
      (8 32 31 7)
      (7 31 30 6)
   )


   wall tank
   (
      (0 24 29 5)
      (5 29 34 10)
      (10 34 38 14)
      (14 38 43 19)
      (19 43 44 20)
      (20 44 45 21)
      (21 45 46 22)
      (22 46 47 23)
      (23 47 42 18)
      (18 42 37 13)
      (13 37 33 9)
      (9 33 28 4)
      (4 28 27 3)
      (3 27 26 2)
      (2 26 25 1)
      (1 25 24 0)
   )

   empty front
   (
      (24 25 30 29)
      (25 26 31 30)
      (26 27 32 31)
      (27 28 33 32)
      (29 30 35 34)
      (32 33 37 36)
      (34 35 39 38)
```

```
        (36 37 42 41)
        (38 39 44 43)
        (39 40 45 44)
        (40 41 46 45)
        (41 42 47 46)
    )

    empty back
    (
            (0 5 6 1)
        (1 6 7 2)
        (2 7 8 3)
        (3 8 9 4)
        (5 10 11 6)
        (8 12 13 9)
        (10 14 15 11)
        (12 17 18 13)
        (14 19 20 15)
        (15 20 21 16)
        (16 21 22 17)
        (17 22 23 18)
    )
);
// ************************************************************************* //
```

Appendix 2
Velocity setting on boundaries at time 0

```
FoamFile
{
   version    2.0;
   format     ascii;
   class      volVectorField;
   object     U;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions     [0 1 -1 0 0 0 0];

internalField   uniform (0 0 0);

boundaryField
{
   front
   {
      type          empty;
   }
   back
   {
      type          empty;
   }
   box
   {
      type          movingWallVelocity;
      value         uniform (0 0 0);
   }
   tank
   {
      type          movingWallVelocity;
      value         uniform (0 0 0);
   }
}
```

Appendix 3
Pressure setting on boundaries at time 0

```
FoamFile
{
   version    2.0;
   format     ascii;
   class      volScalarField;
   object     p;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

dimensions     [1 -1 -2 0 0 0 0];

internalField   uniform 0;

boundaryField
{
   box
   {
      type          buoyantPressure;
      value         uniform 0;
   }

   tank
   {
      type          buoyantPressure;
      value         uniform 0;
   }

   front
   {
      type          empty;
   }
   back
   {
      type          empty;
   }
}
```

Appendix 4
alpha1 setting on boundaries at time 0

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       volScalarField;
    object      alpha1;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //
dimensions      [0 0 0 0 0 0 0];
internalField   uniform 0;
boundaryField

{
    box
    {
        type        zeroGradient;
    }
    tank
    {
        type        zeroGradient;
    }
    front
    {
        type        empty;
    }
    back
        {
        type        empty;
        }
}
```

Appendix 5
alpha1 field setting using “boxToCell” application

```
defaultFieldValues
(
    volScalarFieldValue alpha1 0
);

regions
(
    boxToCell
    {
        box ( -0.5 -5 -5 ) ( 0.5 5 0 );
        fieldValues
        (
            volScalarFieldValue alpha1 1
        );
    }
);
```