Multiphase Flow Simulations

UK FOAM/OpenFOAM User Day

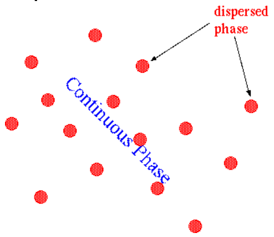
Dr Gavin Tabor

13th April 2015

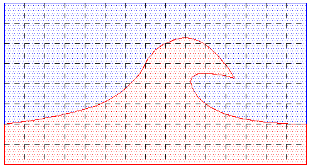
Types of Multiphase Flow

Multiphase flow – 2 (or more) imiscible components occupying the domain and interacting with each other.

Dispersed multiphase flow – sub-grid scale droplets (particles, bubbles) scattered throughout primary continuous phase



Free surface flow – macroscopic boundary between immiscible phases whose location is to be determined



Free Surface Flow

Numerous techniques available in literature; generally divide into

Interface tracking – interface is a computational structure which is moved around

Interface capturing – phases represented by a continuous field, location of interface is derived from this

Interface capturing schemes include variants of *Volume-of-fluid* (VOF) and level set methods.

OpenFOAM contains an implementation of VOF (significant work has been done on method using OF) – codes with `inter` in names

VOF

In VOF, fluid flow (air *or* water) described in terms of a single unified velocity $\underline{u}$ which obeys the momentum equation

$$\frac{\partial \rho \underline{u}}{\partial t} + \nabla . \rho \underline{u}\ \underline{u} = -\nabla p - \underline{g}.\underline{x} \nabla \rho + \nabla . (\mu + \mu_t) \left(\nabla \underline{u} + \nabla \underline{u}^t \right)$$

Phase indicated in terms of *indicator function* α :

$$\alpha = \begin{cases} 0 & \text{in air} \\ 1 & \text{in water} \end{cases}$$

for which

$$\mu = \alpha \mu_{water} + (1 - \alpha) \mu_{air}$$

Interface location is where $0 < \alpha < 1$; however this is spread over 3-4 cells. VOF schemes include mechanisms to sharpen this region; OF implements an artificial velocity field $\underline{w}$ directed normal and towards the interface

$$\frac{\partial \alpha}{\partial t} + \nabla . \underline{u}\alpha + \nabla . \underline{w}(\alpha(1-\alpha)) = 0$$

Pressure includes dynamic and hydrostatic components :

$$P = \rho \underline{g} . \underline{x} + p$$

Field `p_rgh` in the simulation is the dynamic pressure p

Uses MULES (Multi-dimensional Universal Limiter with Explicit Solution) which limits the flux of variables to guarantee bounded solution for α – however this is an explicit scheme (Courant number limitation)

VOF codes

Code	Comments
`interFoam`	Basic VOF code; incompressible phases
`interDyMFoam`	VOF code with mesh motion
`interMixingFoam`	One phase composed of two miscible fluids.
`porousInterFoam`	Explicit handling of porous zones
`interPhaseChangeFoam`	VOF solver including phase change modelling (developed for cavitation but could be extended).
`compressibleInterFoam`	Individual phases compressible

Changes – v2.3.x

No changes to existing code names – some rearrangement (`interMixingFoam` now sub-directory of `interFoam`)

Introduced semi-implicit MULES (combines MULES limiter with explicit correction) – claims to provide boundedness at arbitrary Courant number.

New codes :

`multiphaseInterFoam` – VOF solver for *n* immiscible fluids

`potentialFreeSurfaceFoam` – single plase NSE solver with fluid height function

Dispersed 2nd phase

There are a number of possible approaches to solve systems with dispersed 2nd (3rd, 4th etc.) phases.

Mixture models in which flow is described by a *single* set of NSE (properties such as viscosity represented as mixture); model relative velocity of dispersed phase (slip velocity). Examples – `settlingFoam`

Eulerian multiphase models – each individual phase described by its own set of NSE equations. Interface terms represent forces such as drag, lift. Example – `twoPhaseEulerFoam`

Up to v2.2 – substantial elements of physics hardcoded (eg. turbulence models); some (eg. drag forces) run-time selectable.

Euler-Euler solution

Both the continuous phase and dispersed phase(s) obey NSE; we introduce *conditional averaging* to derive these. Introduce an *indicator function* γ_a :

$$\gamma_a = \begin{cases} 1 & \text{in phase a} \\ 0 & \text{otherwise} \end{cases} \qquad \alpha\overline{\phi}_a = \frac{1}{N}\sum_N \gamma\phi$$

Apply this averaging to the NSE, eg. momentum equation :

$$\frac{\partial \alpha \overline{u}_a}{\partial t} + \nabla.\alpha\overline{u}_a\overline{u}_a + \nabla.\alpha\overline{u'_a u'}_a$$
$$= -\frac{1}{\rho_a}\nabla\overline{p}_a + \nu\nabla^2\alpha\overline{u}_a + \text{interface terms}$$

This provides a momentum equation for each individual phase.

Interface terms

These represent the physics of the interaction between the phases – primarily momentum exchange; eg. lift, drag, VM etc.

In `twoPhaseEulerFoam`, some of this is hard coded; some run-time selectable through dictionaries :

`interfacialProperties` – select drag model for both phases

`kineticTheoryProperties` – range of models describing stresses in solid particulate phase (if necessary)

`ppProperties` – particle-particle interaction forces; activate by setting g0$>$ 0; solids volume fraction limited to `alphaMax` packing limit value

Interface terms (cont)

Drag forces described in terms of drag coefficient K :

$$\underline{f} = K\underline{u}_r$$

K can be related to the drag coefficient for the discrete particles,

$$K = \frac{3}{4} C_d \rho_B \frac{u_r}{d_A}$$

e.g. Schiller-Naumann drag model :

$$C_d = \begin{cases} \frac{24}{\mathcal{R}e}\left(1 + 0.15\mathcal{R}e^{0.687}\right) & \text{for } \mathcal{R}e < 1000 \\ 0.44 & \mathcal{R}e \geq 1000 \end{cases}$$

Interface terms (cont)

Drag models include `Ergun`, `Gibilaro`, `GidaspowErgunWenYu`, `GidaspowSchillerNaumann`, `SchillerNaumann`, `SymlalOBrien`, `WenYu`

Naming conventions (and much of the modelling) follows :

"Eulerian two-phase flow theory applied to fluidization", Enwald, Peirano, Almstedt. *Int.J.Multiphase Flow* **22 Supp** pp.21 – 66 (1996)

Lift and virtual mass forces also included

$$\underline{f}_L = C_L \rho V_B \underline{u}_r \times \nabla \times \underline{u} \quad , \qquad \underline{f}_{VM} = \beta \frac{\rho_B}{\rho_A} C_{VM} \left(\frac{D_B \underline{u}_B}{Dt} - \frac{D_A \underline{u}_A}{Dt} \right)$$

No modelling necessary – just specify coefficients as appropriate

Dispersed multiphase codes

Code	Comments
`bubbleFoam`	Original Euler-euler solver for two incompressible phases
`twoPhaseEulerFoam`	More generic Euler-euler solver (can include solid particulate phase)
`compressibleTwoPhaseEulerFoam`	Solver for compressible fluids (includes heat transfer)
`multiphaseEulerFoam`	Solver for *n* compressible fluid phases
`settlingFoam`	Incompressible mixture model for sedimentation
`twoLiquidMixingFoam`	Mixture model for two incompressible fluids

OpenFOAM solvers – v.2.3

All phases considered to be compressible (thermodynamic *p*; include thermophysical modelling)

Introduce *phase property naming* – consistent formal approach to naming files. `phaseProperties` dictionary contains an entry of the form

```
phases (air water);
```

which specifies the names of the phases involved; all quantities are denoted

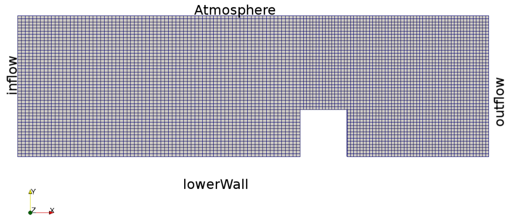
```
<property>.<phase name> eg alpha.water
```

Codes now use standard turbulence modelling framework – allows use of standard templated turbulence models (allows compressible and incompressible phase turbulence)

Case – broad crested weir

Regular rectangular mesh – 5 blocks defined with `blockMesh`

Air `alpha=0`; water `alpha=1`.



Need to initialise water behind weir (`setFields`) and inlet b.c. for `alpha` (use `groovyBC` from `swak4foam`). Need to include

```
libs (``libgroovyBC.so'')
```

in `controlDict`

Initialisation

Need to generate a block of water behind the weir – use `setFields` utility and `system/setFieldsDict`

```
defaultFieldValues
(
    volScalarFieldValue alpha1 0
);

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

Boundary Conditions – alpha

```
boundaryField
{
    inflow
    {
        type            groovyBC;
        variables       "surface=2.0;";
        valueExpression "(pos().y<=surface)␣?␣1.0␣:␣0.0";
        value           uniform 1.0;
    }
    outflow
    {
        type            inletOutlet;
        inletValue      uniform 0;
        value           uniform 0;
    }
    lowerWall
    {
        type            zeroGradient;
    }
    atmosphere
    {
        type            inletOutlet;
        inletValue      uniform 0;
        value           uniform 0;
    }
}
```

p_rgh

```
boundaryField
{
    outflow
    {
        type                totalPressure;
        p0                  uniform 0;
        U                   U;
        phi                 phi;
        rho                 rho;
        psi                 none;
        gamma               1;
        value               uniform 0;
    }
    lowerWall
    {
        type                buoyantPressure;
        value               uniform 0;
    }
    atmosphere
    {
        type                totalPressure;
        p0                  uniform 0;
        U                   U;
        phi                 phi;
        rho                 rho;
        psi                 none;
        gamma               1;
        value               uniform 0;
```

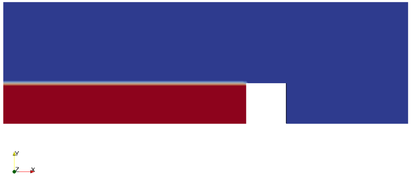
Running

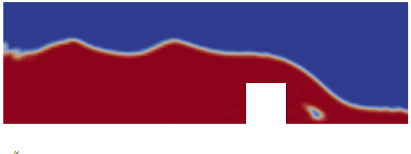
1. Run `blockMesh`
2. Copy `0.orig` to `0`
3. Run `setfields`
4. Run `interFoam`

   ⋮

5. Post-process results

Results





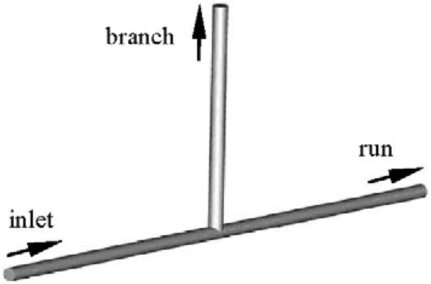
Case – T-junction

Two-phase flow in T-junction important test case – eg:

“Oil-water two-phase Flow inside T-junction”, Wang *et al*, *J.Hydrodynamics* **20(2)** pp. 147 – 153 (2008), “Phase separation of liquid-liquid two-phase flow at a T-junction”, Yang *et al*, *AIChE.J.* **52(1)** pp. 141 – 149 (2006)

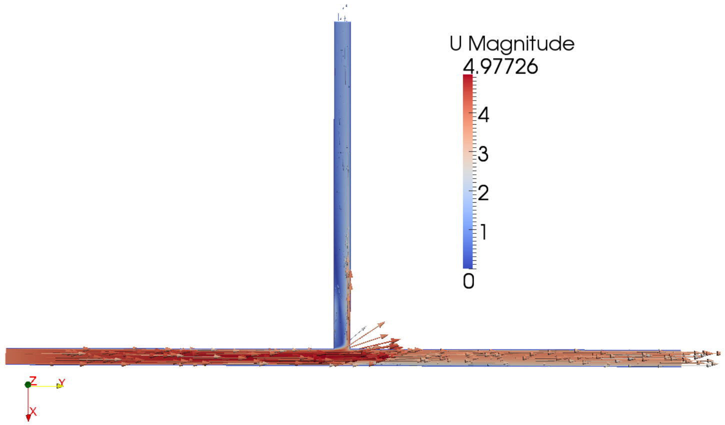
Experimental data for kerosene/water

OpenFOAM has a T-junction tutorial case `Tjunction` for `pimpleFoam` – can be modified to suit



Modify the case

1. Copy the case files to your home directory (obviously!)
2. Modify `blockMeshDict` to give 1m pipes, 5cm across
3. Increase mesh resolution – pipes $20 \times 20 \times 250$
4. Swap over `Inlet` and `Outlet1` patches
5. Adapt inlet conditions :
    - `fixedValue (0 4.0 0)` for `U`
    - `zeroGradient` for `p`
6. Change outlet conditions (`p`) – same pressure at both outlets
7. Re-run to check it still works (reduce $\delta t \rightarrow 0.0001$)


U Magnitude
4.97726
4
3
2
1
0
Z
Y
X

Convertion to 2-phase flow

Need to copy across files from (eg)
`tutorials/multiphase/twoPhaseEulerFoam/bubbleColumn` :

- From `constant` : `g`, `interfacialProperties`, `kineticTheoryProperties`, `ppProperties`, `transportProperties`, `RASProperties`
- From `system` : `fvSchemes`, `fvSolution`

No need to change most of these (`kineticTheoryProperties` basically switched off, `interfacialProperies` set to `SchillerNaumann`)

$$\underline{g} = (9.81 \quad 0 \quad 0)$$

transportProperties

Modify to use physical properties for kerosene and water :

Quantity	Water (phaseb)	Kerosene (phasea)
nu	$1.002 \times 10^{-6}\ \mathrm{m}^2/\mathrm{s}$	$3.1 \times 10^{-5}\ \mathrm{m}^2/\mathrm{s}$
rho	$998\ \mathrm{kg}/\mathrm{m}^3$	$836\ \mathrm{kg}/\mathrm{m}^3$
d	$0.0001\ \mathrm{m}$	$0.003\ \mathrm{m}$

Other quantities – `Cvm` – probably irrelevant – `Cl` – set to 0.5

`fvSchemes` : need to introduce `divSchemes` for `epsilon, k` :

```
laplacian(DepsilonEff,epsilon) Gauss linear corrected; laplacian(DkEff,k)
Gauss linear corrected;
```

transportProperties

```
phaseb                     // water
{
    nu                  nu [ 0 2 -1 0 0 0 0 ] 1.002e-06;
    rho                 rho [ 1 -3 0 0 0 0 0 ] 998;
    d                   d [ 0 1 0 0 0 0 0 ] 0.0001;
}

phasea                     // air
{
    nu                  nu [ 0 2 -1 0 0 0 0 ] 3.1e-05;
    rho                 rho [ 1 -3 0 0 0 0 0 ] 836.0;
    d                   d [ 0 1 0 0 0 0 0 ] 0.003;
}

Cvm             Cvm [ 0 0 0 0 0 0 0 ] 1.0;

Cl              Cl [ 0 0 0 0 0 0 0 ] 0.5;

Ct              Ct [ 0 0 0 0 0 0 0 ] 1;
```

New fields

Several new fields needed in `0` – `alpha`, `Ua`, `Ub`, `Theta` :

- `Ua`, `Ub` – based on existing `U` field – fixed inlet velocities ($0.480\ \mathrm{m/s}$ – `zeroGradient` outlet conditions)

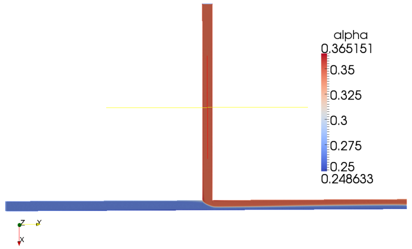
`alpha`, `Theta` – based on `k` (same basic b.c.). Inlet `alpha` 0.2656

`k`, `epsilon` – corrected to more reasonable values ($k = 0.00075\ \mathrm{m^2/s^2}, \epsilon = 0.00035\ \mathrm{m^2/s^3}$)

Also reduced max Courant number `maxCo` to 2.0

Results

	α in branch
Wang *et al*	0.3916
Me	0.3651



But is this correct?

Conclusions

- OpenFOAM codes for multiphase flow :
  - `inter` – VOF solvers
  - `euler` – dispersed multiphase
  - . . . also mixture/slip velocity formulations
- Codes trace back to PhD work of David Hill, Henrik Rusche (dispersed multiphase), Onno Ubbink (VOF)
- Substantial changes in v.2.3 to dispersed multiphase solvers

Contact me for notes/case files – g.r.tabor@ex.ac.uk

References

"The Computer Simulation of Dispersed Two-Phase Flows", D.P.Hill, PhD Thesis, Imperial College of Science, Technology and Medicine (1998)

"Computational Fluid Dynamics of Dispersed Two-Phase Flows at High Phase Fractions", H.Rüsche, PhD Thesis, Imperial College of Science, Technology and Medicine (2002)

"Numerical Prediction of Two Fluid Systems with Sharp Interfaces", O. Ubbink, PhD Thesis, Imperial College of Science, Technology and Medicine (1997)