OpenFOAM Workshop Training Session

Integrated Development Environment (IDE)
Eclipse for OpenFOAM®

Astrid Mahrla[1] and Holger Marschall[2]

[1]Chair of Chemical Engineering, Technische Universität München
[2]Center of Smart Interfaces, Technische Universität Darmstadt
Germany

6th OpenFOAM Workshop
13 - 16 June 2011
The Pennsylvania State University
State College, PA, USA

Outline

1. Introduction
   - What is Eclipse?
   - Scope & Objective
   - Governing Equations & Models
2. Adding a runTime-selective Model – Drag Force
   - Preliminary steps
   - Modifying bubbleFoam
   - Implementing the models
3. Adjusting the case files
4. Debugging with Eclipse
5. Write your own interfacial model!

What is Eclipse?

"Eclipse is an open source community whose projects are focused on building an extensible development platform, runtimes and application frameworks for building, deploying and managing software across the entire software lifecycle. Many people know us, and hopefully love us, as a Java IDE but Eclipse is much more than a Java IDE." – www.eclipse.org

- Integrated Development Environment (IDE) – originally developed for Java programming
- C/C++ Development Toolkit (CDT) extension for *fast and efficient* C++ programming

What is Eclipse?

Features

- Well-arranged graphical user interface offering project explorer, outline, ...
- Fully integrated powerful text editor offering code highlighting, autocompletion, ...
- Integrated compiler offering linked error and warning marks
- Integrated debugger and debugging environment offering breakpoints and variable information
- Project management: bookmarks and tasks
- Extensions: version management, multiple language support (Java, Python, ...)

Scope & Objective

Get to know . . .

- Eclipse development features for applications, libraries, . . .
- Debugging in the Eclipse environment using GNU Debugger (GDB)

. . . restructuring bubbleFoam

- Reorganization of 'hard-coded' interfacial models into C++ libraries that allow runTime-selective access
- Implementation of new interfacial models

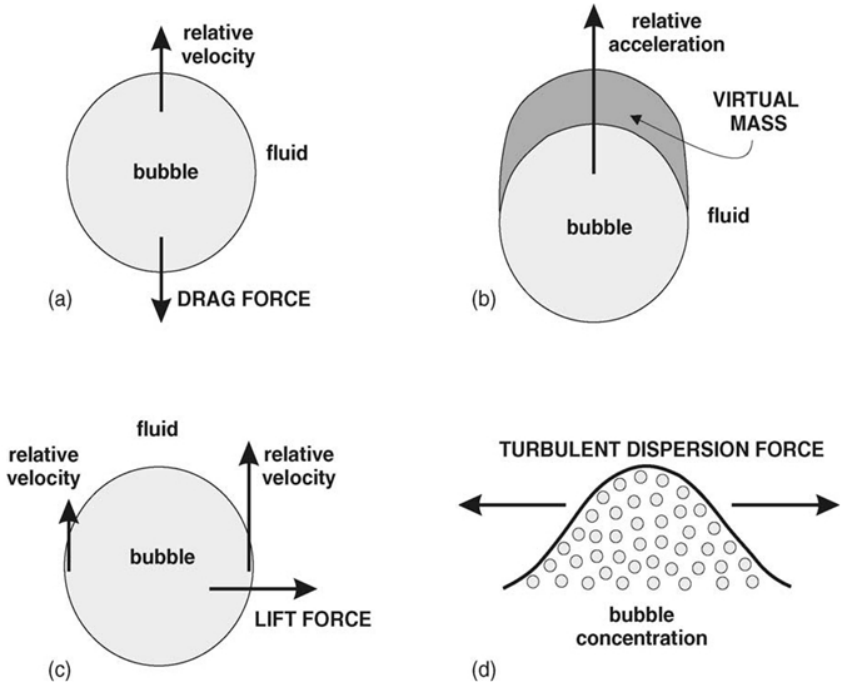
Governing Equations & Models

Two-Fluid-Model – Interfacial Forces

Modelling the interfacial forces is crucial to the simulation of hydrodynamics in bubble columns

- Drag forces (a)
- Non-drag forces
  - Virtual mass force (b)
  - Lateral lift force (c)
  - Turbulent dispersion force (d)

$$\sum \overline{\mathbf{M}}_{\varphi} = \mathbf{F}_D + \mathbf{F}_L + \mathbf{F}_{VM} - \mathbf{F}_{TD}$$



Mendez **2006**

Governing Equations & Models

Two-Fluid-Model – Interfacial Forces

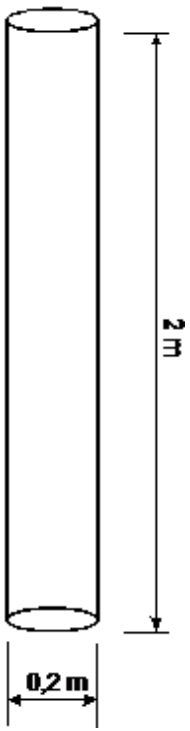
- Drag force
  $\mathbf{F}_D = \frac{3}{4} \cdot \alpha \cdot C_D \cdot \frac{\rho_b}{d_a} \cdot (\mathbf{U}_b - \mathbf{U}_a) \cdot |U_b - U_a|$

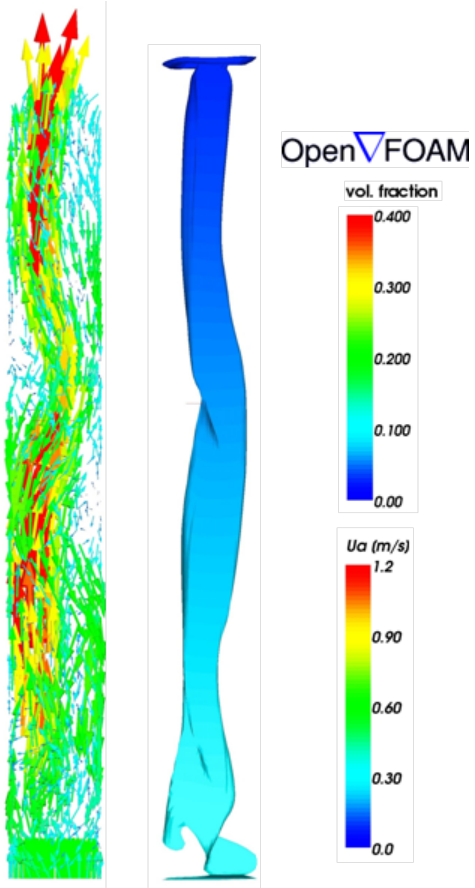
- Non-drag forces
  - Virtual mass force
    $\mathbf{F}_{VM} = \alpha \cdot C_{VM} \cdot \rho_b \cdot \left(\frac{D_b \mathbf{U}_b}{Dt} - \frac{D_a \mathbf{U}_a}{Dt} \right)$ where $\left(\frac{D_i}{Dt} = \frac{\partial}{\partial t} + \mathbf{U}_i \cdot \nabla \right)$
  - Lateral lift force
    $\mathbf{F}_L = \alpha \cdot C_L \cdot \rho_b \cdot (\mathbf{U}_b - \mathbf{U}_a) \times \omega_b$ where $\omega_b = \nabla \times \mathbf{U}_b$
  - Turbulent dispersion force
    $\mathbf{F}_{TD} = C_{TD} \cdot \rho_b \cdot k \cdot \nabla \alpha$

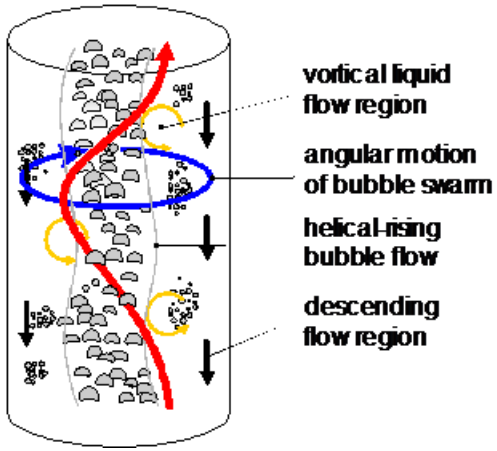
Mornhinweg **2008**, Kossmann **2007**

Simulation – Two-Fluid-Model

Numerical results – Simulation of a cylindric bubble column







Volumetric phase fraction & velocity of the gas phase ($j_g = 0.06$ m/s, $\Delta t = 2.0$ s)

Adding a runTime-selective Model – Drag Force

Preliminary steps

- Start `OpenFOAM-1.6-ext-dbg`-terminal
- `export PATH=/home/ubuntu/gdb4Eclipse/bin:$PATH`
- Create a personal version of `bubbleFoam`
- Copy `cylindricBubbleColumn` test case
- Rename `bubbleFoam` to `bubbleFoamMod`
- Run `blockMesh` on the `cylindricBubbleColumn`

bubbleFoam solver

/usr/lib/OpenFOAM-1.6-ext-dbg/applications/solvers/multiphase/bubbleFoam

cylindricBubbleColumn test case

/cdrom/OFW6/Training/MahrlaMarschall-OFW6_material_Eclipse4OpenFOAM.tgz

Adding a runTime-selective Model – Drag Force

Modifying bubbleFoam – Setting up Eclipse for OpenFOAM

- Launch Eclipse: `eclipse &`
- Choose default workspace
- Open C++ view: *Window* → *Open Perspective* → *Other..* → *C/C++ Projects*
- *Un*check *Project* → *Build Automatically*

Import of `bubbleFoam` solver and set compiler properties

- *Files* → *New* → *C++-Project*
- Deactivate *Use default location* and choose `bubbleFoamMod` solver (*project name*: `bubbleFoamMod`)
- Right-click on project → *Properties* → *C/C++ Build*
- Default build command: `wmake`
- Deactivate *Generate Makefiles automatically*
- Remove `/Debug` in build directory

Adding a runTime-selective Model – Drag Force

Modifying bubbleFoam – Developing and compiling with Eclipse

- Open `bubbleFoam.C`
- *Right-click* → *Preferences . . .* → *Editor*
- Click on *Text Editors* and check *Show line numbers*

Adapt `bubbleFoamMod` solver

- Rename `bubbleFoam.C` to `bubbleFoamMod.C` *(F2)*
- Change `Make/files`
  – don't forget `EXE = $(FOAM_USER_APPBIN)/bubbleFoamMod`

Recompile the new solver by adding new make targets

- Select *Make Targets* tab – select `bubbleFoamMod`-folder
- Click on *New Make Target* button
- Target name: `wmake` – Make target: `<empty>` – Build command: `wmake`
  Target name: `wclean` – Make target: `<empty>` – Build command: `wclean`
- Execute *make targets* by double-clicking

Adding a runTime-selective Model – Drag Force

Modifying bubbleFoam – Altering bubbleFoam (1)

twoPhaseEulerFoam

/usr/lib/OpenFOAM-1.6-ext-dbg/applications/solvers/multiphase/twoPhaseEulerFoam with subfolders /interfacialModels and /phaseModel

Copy `interfacialModels` and `phaseModel` from `twoPhaseEulerFoam`

- In terminal: copy `/interfacialModels` and `/phaseModel` into `/bubbleFoamMod`
- In Eclipse: Refresh workspace *File* → *Refresh (F5)*
- Delete all subdirectories in `interfacialModels/dragModels` except `/dragModel` and `/SchillerNaumann`

Adapt corresponding `Make`-files

- Adapt `/interfacialModels/Make/files` and `options` – delete unused drag models *(Source 1 and 2)*
- Adapt `/phaseModel/Make/files` *(Source 3)*
- Adapt `/bubbleFoamMod/Make/options` *(Source 4)*

Adding a runTime-selective Model – Drag Force

Modifying bubbleFoam – Altering bubbleFoam (2)

Embed new models in `createFields.H`

- Create phase model auto pointers for each phase *(Source 5)*
  `autoPtr<phaseModel> phasea = phaseModel::New(..);`
- Read out phase properties `rho()`, `nu()`, `d()` using the `->` operator *(Source 5)*
  `const dimensionedScalar& rhoa = phasea->rho();`
- Create new dictionary `interfacialProperties` *(Source 6)*
  `IOdictionary interfacialProperties (..);`
- Create drag model auto pointers for each phase *(Source 6)*
  `autoPtr<dragModel> draga = dragModel::New(..);`

Further top level changes

- `#include dragModel.H` in `bubbleFoamMod.C` *(Source 7)*
- Adapt `liftDragCoeffs.H` *(Source 8)*

Adding a runTime-selective Model – Drag Force

Modifying bubbleFoam – Altering bubbleFoam (3)

Library compilation in Eclipse

- Select *Make Targets* tab – select `interfacialModels`
- Click on *New Make Target* button
- Target name: `wmake libso` – Make target: `libso` – Build command: `wmake`
  Target name: `wclean libso` – Make target: `libso` – Build command: `wclean`
- Create make targets for `phaseModel` library and compile it
- Compile `interfacialModels` library
- Compile `bubbleFoamMod` solver

Adding a runTime-selective Model – Drag Force

Implementing the models

Implementation of Tomiyama drag model

- Create `Tomiyama98` directory in `/dragModels`
- Copy all `SchillerNaumann` files to `Tomiyama98`
- Rename to `Tomiyama98.C` and `Tomiyama98.H` *(F2)*
- Add Tomiyama model in `interfacialModels/Make/files`: `dragModels/Tomiyama98/Tomiyama98.C`
- Change `Tomiyama98.C` and `Tomiyama98.H` *(Source 9 and 10)* change the name: *Edit* → *Find/Replace*

Read out surface tension and gravity from phase model

- Add variables to dictionary in `phaseModel.C` *(Source 11)*
- Add variables and access functions in `phaseModel.H` *(Source 12)*

Recompilation

- Recompile both libraries and the solver

Adjusting the case files

Simulation run in Eclipse

- Import `cylindricBubbleColumn` test case
- Choose the drag force model in `constant/interfacialProperties`

Set up run configurations

- *Run* → *Run Configurations*
- Project: `cylindricBubbleColumn` – application: `bubbleFoamMod` (debug binary!)
- Click *Run* button

Debugging with Eclipse

Access to runTime-selective models

Set up debug configurations

- *Run* → *Debug Configurations* – Choose same configurations as for run
- Check the use of GDB (DSF) Create Process Launcher as debugger
- Click the *Debug* button

Set breakpoints

- 1st breakpoint: `createFields.H`
  `autoPtr<dragModel> draga = dragModel::New(..);`
- 2nd breakpoint: `liftDragCoeffs.H`
  `volScalarField Cda = draga->K(magUr);`

Access of runTime-selective models

- Resume to 1st breakpoint *(F8)*
- Step into drag model selection process *(F5)*
- Resume to 2nd breakpoint *(F8)*
- Step into the evaluation of the drag coefficient *(F5)*
- Step into the evaluation of Reynolds number *(F5)*
  `volScalarField Re = max(Ur*phasea_.d()/phaseb_.nu(), 1.0e-03);`

Use *F6* to step *over* functions!

Write your own interfacial model!

Implement further interfacial models!

You are invited to join a collaborate project on enhancing `bubbleFoam` towards a full set of state-of-the-art closure equations for bubble force.

Join www.extend-project.de and click on Community-driven Collaborative Projects in the left menu panel!

Keep foaming!