MultiFlow: A coupled balanced-force framework to solve multiphase flows in arbitrary domains

Berend van Wachem & Fabien Evrard

Fakultät für Verfahrens- und Systemtechnik
Institut für Verfahrenstechnik

berend.vanwachem@ovgu.de
fabien.evrard@ovgu.de
mvt.ovgu.de

2023 Annual PETSc Meeting
Chicago, June 5–7

Motivation

- Multiphase flows are frequently encountered in industry and nature.

Motivation

- Multiphase flows are frequently encountered in industry and nature.



- The equations governing these flows may involve:
  - volume fraction and source term gradients
  - property discontinuities
  - multiple sets of velocities
  - etc.

Motivation

- Multiphase flows are frequently encountered in industry and nature.



- The equations governing these flows may involve:
  - volume fraction and source term gradients
  - property discontinuities
  - multiple sets of velocities
  - etc.
- Current algorithms for multiphase flows are typically based on single phase flows.

Motivation

- Multiphase flows are frequently encountered in industry and nature.



- The equations governing these flows may involve:
  - volume fraction and source term gradients
  - property discontinuities
  - multiple sets of velocities
  - etc.
- Current algorithms for multiphase flows are typically based on single phase flows.
  - They lack efficiency and robustness for multiphase flows.

Motivation

- Multiphase flows are frequently encountered in industry and nature.



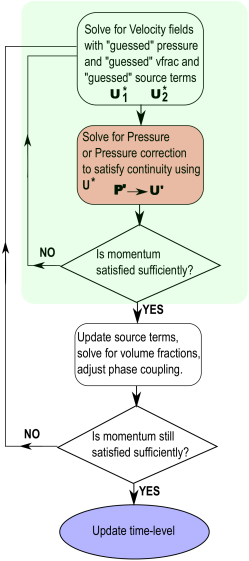
- The equations governing these flows may involve:
  - volume fraction and source term gradients
  - property discontinuities
  - multiple sets of velocities
  - etc.
- Current algorithms for multiphase flows are typically based on single phase flows.
  - They lack efficiency and robustness for multiphase flows.
- Most flows occur in complex geometries, therefore a collocated variable arrangement is more natural (in a finite volume framework).

Introduction

Incompressible Navier-Stokes equations

$$\begin{aligned} \nabla \cdot \boldsymbol{u} &= 0 \\ \rho \left[\frac{\partial \boldsymbol{u}}{\partial t} + \nabla \cdot (\boldsymbol{u} \otimes \boldsymbol{u}) \right] &= -\nabla p + \nabla \cdot \boldsymbol{\tau} + \boldsymbol{s} \end{aligned}$$

- The majority of finite-volume algorithms for incompressible flows are based on a segregated solver approach:

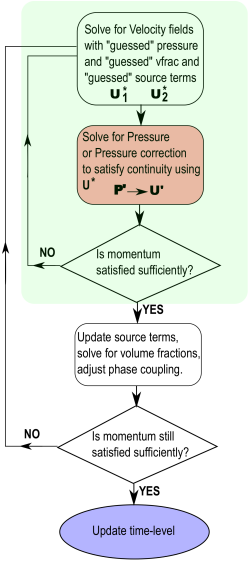


Introduction

Incompressible Navier-Stokes equations

$$\begin{aligned} \nabla \cdot \boldsymbol{u} &= 0 \\ \rho \left[\frac{\partial \boldsymbol{u}}{\partial t} + \nabla \cdot (\boldsymbol{u} \otimes \boldsymbol{u}) \right] &= -\nabla p + \nabla \cdot \boldsymbol{\tau} + \boldsymbol{s} \end{aligned}$$

- The majority of finite-volume algorithms for incompressible flows are based on a segregated solver approach:
  1. Estimate velocities with momentum equation with guessed pressure, volume fractions, etc.

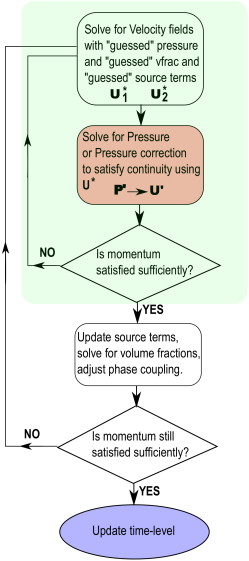


Introduction

Incompressible Navier-Stokes equations

$$\begin{aligned} \nabla \cdot \boldsymbol{u} &= 0 \\ \rho \left[\frac{\partial \boldsymbol{u}}{\partial t} + \nabla \cdot (\boldsymbol{u} \otimes \boldsymbol{u}) \right] &= -\nabla p + \nabla \cdot \boldsymbol{\tau} + \boldsymbol{s} \end{aligned}$$

- The majority of finite-volume algorithms for incompressible flows are based on a segregated solver approach:
  1. Estimate velocities with momentum equation with guessed pressure, volume fractions, etc.
  2. Correct velocity field with continuity equation through pressure.

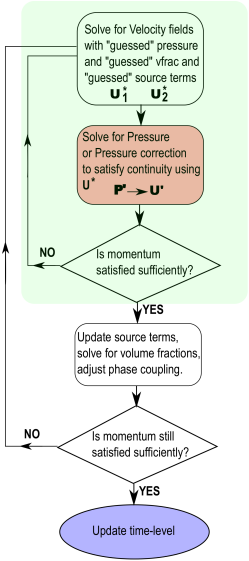


Introduction

Incompressible Navier-Stokes equations

$$\begin{aligned} \nabla \cdot \boldsymbol{u} &= 0 \\ \rho \left[\frac{\partial \boldsymbol{u}}{\partial t} + \nabla \cdot (\boldsymbol{u} \otimes \boldsymbol{u}) \right] &= -\nabla p + \nabla \cdot \boldsymbol{\tau} + \boldsymbol{s} \end{aligned}$$

- The majority of finite-volume algorithms for incompressible flows are based on a segregated solver approach:
  1. Estimate velocities with momentum equation with guessed pressure, volume fractions, etc.
  2. Correct velocity field with continuity equation through pressure.
     → *Requires to solve a **Poisson equation**.*

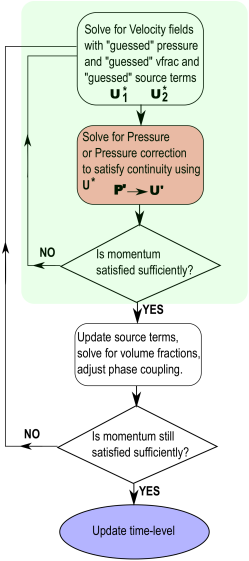


Introduction

Incompressible Navier-Stokes equations

$$\begin{aligned} \nabla \cdot \boldsymbol{u} &= 0 \\ \rho \left[\frac{\partial \boldsymbol{u}}{\partial t} + \nabla \cdot (\boldsymbol{u} \otimes \boldsymbol{u}) \right] &= -\nabla p + \nabla \cdot \boldsymbol{\tau} + \boldsymbol{s} \end{aligned}$$

- The majority of finite-volume algorithms for incompressible flows are based on a segregated solver approach:
  1. Estimate velocities with momentum equation with guessed pressure, volume fractions, etc.
  2. Correct velocity field with continuity equation through pressure.
     → *Requires to solve a* ***Poisson equation****.*
  3. Update volume fractions, source terms, gradients, etc.

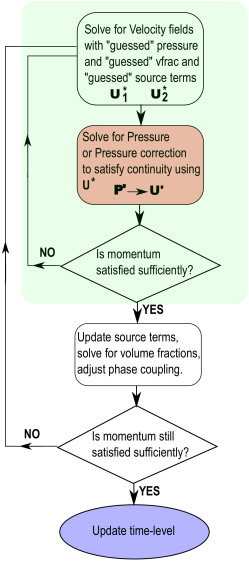


Introduction

Incompressible Navier-Stokes equations

$$\begin{aligned} \nabla \cdot \boldsymbol{u} &= 0 \\ \rho \left[\frac{\partial \boldsymbol{u}}{\partial t} + \nabla \cdot (\boldsymbol{u} \otimes \boldsymbol{u}) \right] &= -\nabla p + \nabla \cdot \boldsymbol{\tau} + \boldsymbol{s} \end{aligned}$$

- The majority of finite-volume algorithms for incompressible flows are based on a segregated solver approach:
  1. Estimate velocities with momentum equation with guessed pressure, volume fractions, etc.
  2. Correct velocity field with continuity equation through pressure.
     → *Requires to solve a* ***Poisson equation****.*
  3. Update volume fractions, source terms, gradients, etc.
  4. Make sure phases are sufficiently coupled.

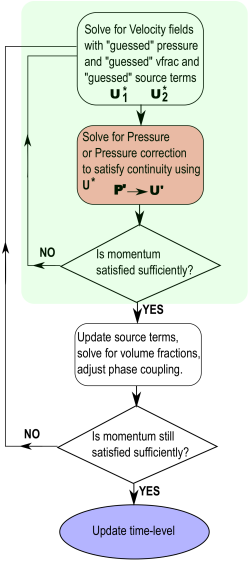


Introduction

Incompressible Navier-Stokes equations

$$\begin{aligned} \nabla \cdot \boldsymbol{u} &= 0 \\ \rho \left[\frac{\partial \boldsymbol{u}}{\partial t} + \nabla \cdot (\boldsymbol{u} \otimes \boldsymbol{u}) \right] &= -\nabla p + \nabla \cdot \boldsymbol{\tau} + \boldsymbol{s} \end{aligned}$$

- The majority of finite-volume algorithms for incompressible flows are based on a segregated solver approach:
  1. Estimate velocities with momentum equation with guessed pressure, volume fractions, etc.
  2. Correct velocity field with continuity equation through pressure.
     → *Requires to solve a* ***Poisson equation****.*
  3. Update volume fractions, source terms, gradients, etc.
  4. Make sure phases are sufficiently coupled.
  5. Go back to 1.



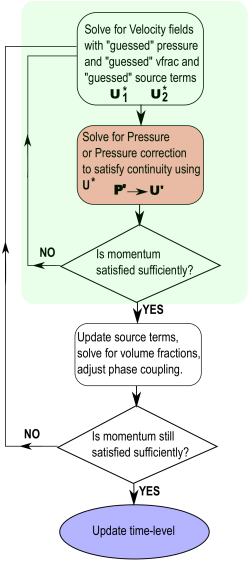
Introduction

Incompressible Navier-Stokes equations

$$\begin{aligned} \nabla \cdot \boldsymbol{u} &= 0 \\ \rho \left[\frac{\partial \boldsymbol{u}}{\partial t} + \nabla \cdot (\boldsymbol{u} \otimes \boldsymbol{u}) \right] &= -\nabla p + \nabla \cdot \boldsymbol{\tau} + \boldsymbol{s} \end{aligned}$$

- The majority of finite-volume algorithms for incompressible flows are based on a segregated solver approach:
  1. Estimate velocities with momentum equation with guessed pressure, volume fractions, etc.
  2. Correct velocity field with continuity equation through pressure.
     → *Requires to solve a* ***Poisson equation****.*
  3. Update volume fractions, source terms, gradients, etc.
  4. Make sure phases are sufficiently coupled.
  5. Go back to 1.

▷ This requires **underrelaxation**.



Introduction

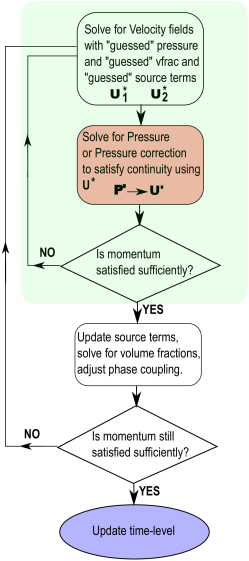
Incompressible Navier-Stokes equations

$$\begin{aligned} \nabla \cdot \boldsymbol{u} &= 0 \\ \rho \left[\frac{\partial \boldsymbol{u}}{\partial t} + \nabla \cdot (\boldsymbol{u} \otimes \boldsymbol{u}) \right] &= -\nabla p + \nabla \cdot \boldsymbol{\tau} + \boldsymbol{s} \end{aligned}$$

- The majority of finite-volume algorithms for incompressible flows are based on a segregated solver approach:
  1. Estimate velocities with momentum equation with guessed pressure, volume fractions, etc.
  2. Correct velocity field with continuity equation through pressure.
     → *Requires to solve a* ***Poisson equation****.*
  3. Update volume fractions, source terms, gradients, etc.
  4. Make sure phases are sufficiently coupled.
  5. Go back to 1.

▷ This requires **underrelaxation**.

▷ There is **no guarantee** for a solution.



Introduction

Incompressible Navier-Stokes equations

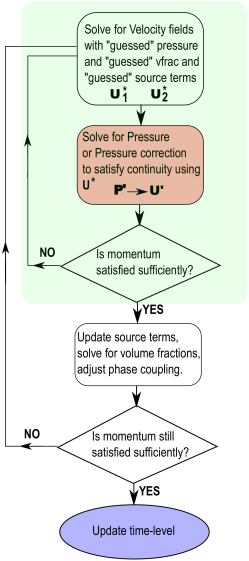
$$\begin{aligned} \nabla \cdot \boldsymbol{u} &= 0 \\ \rho \left[\frac{\partial \boldsymbol{u}}{\partial t} + \nabla \cdot (\boldsymbol{u} \otimes \boldsymbol{u}) \right] &= -\nabla p + \nabla \cdot \boldsymbol{\tau} + \boldsymbol{s} \end{aligned}$$

- The majority of finite-volume algorithms for incompressible flows are based on a segregated solver approach:
  1. Estimate velocities with momentum equation with guessed pressure, volume fractions, etc.
  2. Correct velocity field with continuity equation through pressure.
     → *Requires to solve a* ***Poisson equation****.*
  3. Update volume fractions, source terms, gradients, etc.
  4. Make sure phases are sufficiently coupled.
  5. Go back to 1.

▷ This requires **underrelaxation**.

▷ There is **no guarantee** for a solution.

▷ Difficulties arise with **large source terms**.



Introduction

- Strategies for coupled solving (instead of segregated):

Introduction

- Strategies for coupled solving (instead of segregated):
  - ▷ Accepting a zero on the diagonal of the linearized matrix.
    - → Requires a fancy solver.

Introduction

- Strategies for coupled solving (instead of segregated):
  - ▷ Accepting a zero on the diagonal of the linearized matrix.
    - → Requires a fancy solver.
  - ▷ Artificial compressibility.
    - → Difficult with two phases, volume fractions and source terms.

Introduction

- Strategies for coupled solving (instead of segregated):
  - ▷ Accepting a zero on the diagonal of the linearized matrix.
    - → Requires a fancy solver.
  - ▷ Artificial compressibility.
    - → Difficult with two phases, volume fractions and source terms.
  - ▷ Including pressure dependency in the continuity equation.
    - → Requires a different approach to the discretisation.

Introduction

- Strategies for coupled solving (instead of segregated):
  - ▷ Accepting a zero on the diagonal of the linearized matrix.
    - → Requires a fancy solver.
  - ▷ Artificial compressibility.
    - → Difficult with two phases, volume fractions and source terms.
  - ▷ **Including pressure dependency in the continuity equation.**
    - → **Requires a different approach to the discretisation.**

Staggered vs. collocated discretisation

- A staggered variable storage mitigates pressure velocity decoupling.
  Perot (2000), Wenneker et al (2003)

▷ The natural discretisation of the pressure gradient directly drives the velocity.

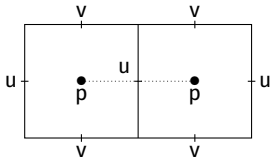
→ *This provides a "natural" coupling between pressure and velocity.*

▷ Very compact stencil for pressure.

▷ Preferred configuration for Cartesian grids.

→ *Used by most research codes.*

▷ For complex geometries . . . not so simple!



Staggered vs. collocated discretisation

- A staggered variable storage mitigates pressure velocity decoupling.
  Perot (2000), Wenneker et al (2003)

▷ The natural discretisation of the pressure gradient directly drives the velocity.

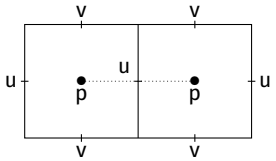
→ *This provides a "natural" coupling between pressure and velocity.*

▷ Very compact stencil for pressure.

▷ Preferred configuration for Cartesian grids.

→ *Used by most research codes.*

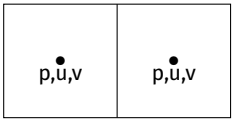
▷ For complex geometries . . . not so simple!



- A collocated variable storage allows for arbitrary meshes.

▷ Trivial application to arbitrary meshes.

▷ But: naive implementation can lead to pressure-velocity decoupling.



Momentum weighted interpolation

- Introduces a pressure dependency in the continuity equation.
- Yields a strong cell-to-cell pressure-velocity coupling.
- Allows us to solve the governing equations as part of a single linear system.

Momentum weighted interpolation

- Introduces a pressure dependency in the continuity equation.
- Yields a strong cell-to-cell pressure-velocity coupling.
- Allows us to solve the governing equations as part of a single linear system.
- The idea of Momentum Weighted Interpolation was first introduced by Rhie and Chow (1983).
- This idea has been further developed in our work for multiphase flow calculations.

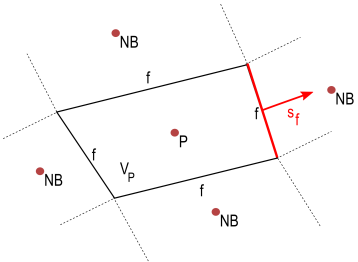
  Denner & van Wachem, *Num. Heat Transfer Part B* 65-3 (2014), 218-255
  Bartholomew et al., *J. Comput. Phys.* 375 (2018), 177-208

Momentum weighted interpolation

Navier-Stokes equations

$$\rho \frac{\partial u_j}{\partial t} + \rho \frac{\partial}{\partial x_i}\left(u_i\, u_j\right) = -\frac{\partial p}{\partial x_j} + \frac{\partial \tau_{ij}}{\partial x_i} - s_j$$

$$\frac{\partial}{\partial x_i} u_i = 0$$



Discretised Navier-Stokes equations

$$\left[\frac{\rho V_P}{\Delta t} + a_P\right] u_{j,P} = \left[\sum_{nb} a_{nb}\, u_{j,nb}\right] - V_P \left[\sum_{nb} b_{nb}\, p_{nb} + s_j\right] + \left[\frac{\rho V_P}{\Delta t}\right] u^O_{j,P}$$

$$\sum_{f=\text{faces}} u_{i,f}\, s_{i,f} = \sum_{f=\text{faces}} \vartheta_f = 0$$

Momentum weighted interpolation

- The net force *driving* the flow is the pressure gradient and sources,

Net driving force

$$\widetilde{\frac{\partial p}{\partial x_j}} = \left[\frac{\partial p}{\partial x_j} - s_j\right]$$

- With this, the discretised equation becomes:

Discretised Momentum equations

$$\left[1 + \frac{\rho}{\Delta t}\frac{V_P}{a_P}\right] u_{j,P} = \left\{\frac{\left[\sum_{nb} a_{nb}\, u_{j,nb}\right]}{a_P}\right\} - V_P \frac{\left[\widetilde{\frac{\partial p}{\partial x_j}}\right]_P}{a_P} + \left[\frac{\rho}{\Delta t}\right]_P \frac{V_P}{a_P} u^O_{j,P}$$

Momentum weighted interpolation

- Using the following

Abbreviations

$$c_P = \frac{\rho}{\Delta t} \qquad d_P = \frac{V_P}{a_P} \qquad \widetilde{u_{j,P}} = \left\{ \frac{[\sum_{nb} a_{nb}\, u_{j,nb}]}{a_P} \right\}$$

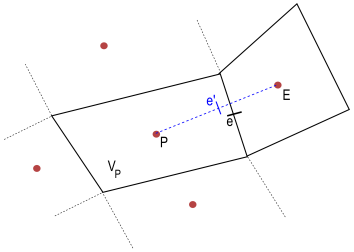
- The discretised equation becomes

Momentum equations

$$[1 + c_P\, d_P]\, u_{j,P} = \widetilde{u_{j,P}} - d_P \left[\widetilde{\frac{\partial p}{\partial x_j}} \right]_P + c_P\, d_P u^O_{j,P}$$

Momentum weighted interpolation

- Such a discretised equation for cell E can also be written out,



- then an *analogous* equation for location e' can be constructed:

Momentum equation at e'

$$[1 + c_{e'}\, d_{e'}]\, u_{j,e'} = \widetilde{u_{j,e'}} - d_{e'} \left[\widetilde{\frac{\partial p}{\partial x_j}} \right]_{e'} + c_{e'}\, d_{e'} u^{O}_{j,e'}$$

Momentum weighted interpolation

- Writing out the terms which are interpolated to e',

U velocity at e'

$$u_{j,e'} = \frac{u_{j,P} + u_{j,E}}{2} - \frac{d_{e'}}{[1 + c_{e'}\, d_{e'}]} \left(\left[\widetilde{\frac{\partial p}{\partial x_j}} \right]_{e'} - \frac{1}{2} \left[\widetilde{\frac{\partial p}{\partial x_j}} \right]_{P} - \frac{1}{2} \left[\widetilde{\frac{\partial p}{\partial x_j}} \right]_{E} \right) + \frac{c_{e'}\, d_{e'}}{[1 + c_{e'}\, d_{e'}]} \left(u^{O}_{j,e'} - \frac{1}{2} u^{O}_{j,P} - \frac{1}{2} u^{O}_{j,E} \right)$$

- There are various ways to go from e' to e, for instance

From e' to e

$$u_{j,e} = u_{j,e'} + \overline{\frac{\partial u_j}{\partial x_i}}_{e'} (x_{i,e'} - x_{i,e})$$

- Now there is an *analogous analytical* expression for the face velocity which depends on pressure.

Momentum weighted interpolation

- This expression can be directly used in the continuity equation.
- $\vartheta = \boldsymbol{u}_f \cdot \boldsymbol{n}_f$, the flux at the face, is only needed.
  - ▷ For a steady-state situation, the expression for ϑ *does not depend on the time-step.*
  - ▷ The pressure terms are analogous to $\Delta^2 \dfrac{\partial^3 p}{\partial x^3}$.
    This is similar to a filter, which converges to zero with the same order as the discretisation.

Momentum weighted interpolation

- This expression can be directly used in the continuity equation.
- $\vartheta = \boldsymbol{u}_f \cdot \boldsymbol{n}_f$, the flux at the face, is only needed.
  - ▷ For a steady-state situation, the expression for ϑ *does not depend on the time-step.*
  - ▷ The pressure terms are analogous to $\Delta^2 \dfrac{\partial^3 p}{\partial x^3}$.
    This is similar to a filter, which converges to zero with the same order as the discretisation.
- The expression can be used in a *finite volume* framework for any type of cell.

Momentum weighted interpolation

- Advected variables discretised with TVD schemes
  Denner & van Wachem, *J. Comput. Phys.* 298 (2015), 466
- Transient terms discretised with backward Euler scheme
  - First-order or second-order backward Euler scheme
  - Same scheme applied for all transient terms

Momentum weighted interpolation

- Advected variables discretised with TVD schemes
  Denner & van Wachem, *J. Comput. Phys.* 298 (2015), 466
- Transient terms discretised with backward Euler scheme
  - First-order or second-order backward Euler scheme
  - Same scheme applied for all transient terms
- Advecting velocity evaluated with momentum-weighted interpolation
  Bartholomew et al., *J. Comput. Phys.* 375 (2018), 177
  - Pressure-velocity coupling for low-Mach flows
  - Source terms require special reconstruction to ensure force balance

Advecting velocity

$$\vartheta_f = \overline{\boldsymbol{u}}_f \boldsymbol{n}_f - \hat{d}_f \left(\boldsymbol{\nabla} p_f - \overline{\boldsymbol{\nabla} p}_f \right) \boldsymbol{s}_f + \hat{d}_f \left(\boldsymbol{S}_f - \overline{\boldsymbol{S}}_f \right) \boldsymbol{s}_f + \hat{d}_f \frac{\rho_f}{\Delta t} \left(\vartheta_f^O - \overline{\boldsymbol{u}}_f^O \boldsymbol{n}_f \right)$$

Solution procedure

- Governing flow equations solved in single equation system
  - Robust even for large pressure or density discontinuities
- No underrelaxation necessary
- Solved using the PETSc library
  - *Block-Jacobi* preconditioner
  - *BiCGStab* solver

Linear system of equations

$$\begin{pmatrix} \boldsymbol{A}_u & \boldsymbol{A}_v & \boldsymbol{A}_w & \boldsymbol{A}_p & \boldsymbol{0} \\ \boldsymbol{B}_u & \boldsymbol{B}_v & \boldsymbol{B}_w & \boldsymbol{B}_p & \boldsymbol{0} \\ \boldsymbol{C}_u & \boldsymbol{C}_v & \boldsymbol{C}_w & \boldsymbol{C}_p & \boldsymbol{0} \\ \boldsymbol{D}_u & \boldsymbol{D}_v & \boldsymbol{D}_w & \boldsymbol{D}_p & \boldsymbol{0} \\ \boldsymbol{E}_u & \boldsymbol{E}_v & \boldsymbol{E}_w & \boldsymbol{E}_p & \boldsymbol{E}_h \end{pmatrix} \cdot \begin{pmatrix} \boldsymbol{\phi}_u \\ \boldsymbol{\phi}_v \\ \boldsymbol{\phi}_w \\ \boldsymbol{\phi}_p \\ \boldsymbol{\phi}_h \end{pmatrix} = \boldsymbol{b}$$

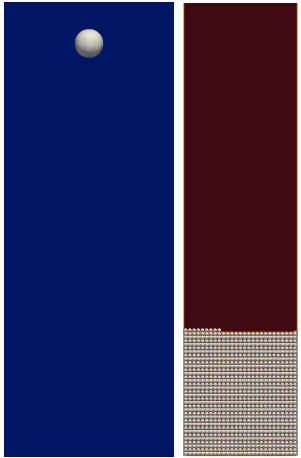
Momentum x
Momentum y
Momentum z
Continuity
Energy

History

- Three versions of *MultiFlow*:
  - All versions are based around the structure of PETSc KSP Solver, Vec, Ghost update, etc.
  1. ***MultiFlow 1*** (since 2004)
     Block structured, static mesh, “one iteration per timestep”.
     - based on multiple interconnected DMDA structures
     - uses PETSc binary file format
     - translation to/from VTK done externally
  2. ***MultiFlow 2*** (since 2013)
     Polyhedral, static mesh.
     - in-house mesh handling routines
     - partitioning with Parmetis
     - HDF5 file format (directly read by Paraview with XDMF reader)
  3. ***MultiFlow 3*** (since 2019)
     Unstructured/block structured/polyhedral, adaptive mesh.
     - Mesh handling based on DMPlex routines.
     - Adaptive refinement based on DMForest/p4est framework.

Examples of applications

Fully resolved particulate flows: IBM simulation



- Conducted in *MultiFlow 2*.
- Solid bodies modelled with IBM.
- $\mathcal{O}(10^3)$ particles.
- No-slip condition enforced with momentum sources in the region surrounding the moving bodies.
- Fully resolved particulate flows but ... very expensive!
  - ▷ Motivation for *MultiFlow 3*.

Examples of applications

Large scale fluidised bed: Euler-Lagrange simulation

- Conducted in *MultiFlow 1*.
- $\mathcal{O}(10^6)$ particles.
- Flow at the scale of the Lagrangian point particles is not resolved.
- Flow and particles are coupled via momentum transfer and volume fraction contribution.
- Individual particle motion solved separately within overlapping Cartesian mesh.

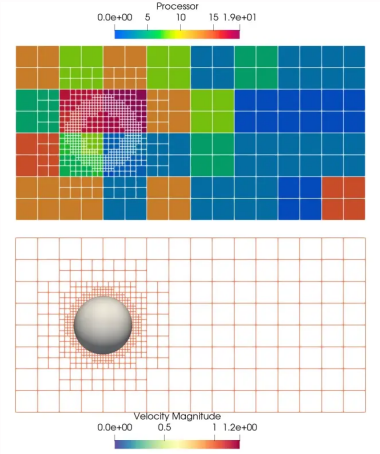
Examples of applications

Atomising swirling spray

- Conducted in *MultiFlow 2*.
- $\mathcal{O}(10^7)$ mesh cells.
- But resolution of all scales would require (much) larger mesh!
  - ▷ Motivation for *MultiFlow 3*.

Examples of applications: MF3

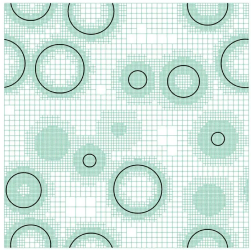
Micro-scale particle laden flow



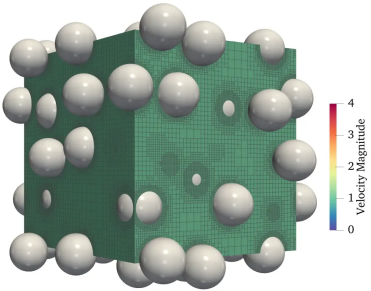
- Conducted in our new *MultiFlow 3*!
- Increasing of resolution where needed with p4est.
- Refinement is based on vorticity.

Examples of applications: MF3

Micro-scale particle laden flow



t = 0.00 U/L



- Conducted in *MultiFlow 3*.
- Moving to $\mathcal{O}(10^3)$ particles and $\mathcal{O}(10^7)$ mesh cells.

MultiFlow 3: implementation

Specifications

We aim to have a coupled framework that:

→ solves the (in)compressible Navier-Stokes equations in the presence of large source term and volume fraction gradients,

→ is designed for arbitrary computational domains,

→ can adapt the mesh where resolution is needed
(*e.g.* at the interface between two fluids, near an immersed boundary)

→ accounts for the specificities of multiphase flow modelling

Used frameworks:

- the DMPlex routines/framework
- the DMForest/p4est framework
- the I/O routines for mesh and data

MultiFlow 3: implementation

DMPlex usage

1. For the “coupled” Vector, we create a `DMPlexCreateSection` with ≥ 4 fields.
2. For the other Vectors, we copy the `DMPlexCreateSection` and set fields to 1.
3. If necessary, we couple the `DMPlex` object to the `DMForest` object with `DMConvert`.
4. For efficiently tracking particles or interfaces, we use a `DMDA` “particle-mesh”.

MultiFlow 3: implementation

Challenges

- We are not computer scientists, and it is hard to understand the details of `DMPlex`
- As we do our own discretisation, some `DMPlex` frameworks are superfluous, but still need to be dealt with (e.g., the `FE` discretisation object).
- Some `DMPlex` implementations do not match our needs, *e.g.* `HDF-5` output and `DMPlexCreateBoxMesh`.
- We have had to implement our own AMR-related routines for:
  - Computing and storing geometric mesh properties.
  - Interpolating from coarse to fine grids.
  - Handling hanging-nodes in the context of finite-volumes.
- We are struggling with the restart of AMR simulations.
- There is a bug in the combination of DMPlex/P4est for periodic meshes.

MultiFlow 3: implementation

More details

- Bartholomew, P., Denner, F., Abdol-Azis, M.H., Marquis, A., van Wachem, B, 2018. Unified formulation of the momentum-weighted interpolation for collocated variable arrangements. Journal of Computational Physics 375, 177-208.
- Denner, F., Evrard, F., van Wachem, B., 2020. Conservative finite-volume framework and pressure-based algorithm for flows of incompressible, ideal-gas and real-gas fluids at all speeds. Journal of Computational Physics 409, 109348.
- Evrard, F., Denner, F., van Wachem, B., 2020. Euler-Lagrange modelling of dilute particle-laden flows with arbitrary particle-size to mesh-spacing ratio. Journal of Computational Physics 8, 100078.
- Cheron, V., Evrard, F., van Wachem, B., 2023. A hybrid immersed boundary method for dense particle-laden flows. Computers & Fluids 105892.
- https://www.mvt.ovgu.de/Publications.html