Graz University of Technology

An OpenFOAM implementation of the filtered two-fluids model

eulerianFilteredTFM Library Architecture

Federico Municchi, Stefan Radl

Institute of Process and Particle Engineering,
TU Graz

Graz, Austria

Unclosed terms in the model eqns

Graz University of Technology

Governing equations

Particle phase:

$$\frac{\partial}{\partial t}\left(\rho_s \overline{\phi}_s \widetilde{\boldsymbol{u}}_s\right) + \boldsymbol{\nabla}\cdot\left(\rho_s \overline{\phi}_s \widetilde{\boldsymbol{u}}_s \widetilde{\boldsymbol{u}}_s\right) = -\overline{\phi}_s \boldsymbol{\nabla}\overline{p} - \boldsymbol{\nabla}\cdot\left(\overline{\boldsymbol{\Sigma}_s^{meso}} + \overline{\boldsymbol{\Sigma}_s^{fric}} + \overline{\boldsymbol{\Sigma}_s^{micro}}\right)$$
$$+ \widetilde{\boldsymbol{f}_{gs}} - \overline{\phi_s' \boldsymbol{\nabla}\cdot\boldsymbol{\sigma}_g'} + \rho_s \overline{\phi}_s \mathbf{g}$$

drag laws accounting for effect of mesoscale structures

3 phenomena to model:
(i) mesoscale,
(ii) frictional, and
(iii) microscale stress

Fluid phase:

$$\frac{\partial}{\partial t}\left(\rho_g \overline{\phi}_g \widetilde{\boldsymbol{u}}_g\right) + \boldsymbol{\nabla}\cdot\left(\rho_g \overline{\phi}_g \widetilde{\boldsymbol{u}}_g \widetilde{\boldsymbol{u}}_g\right) = -\overline{\phi}_g \boldsymbol{\nabla}\overline{p} - \boldsymbol{\nabla}\cdot\left(\overline{\boldsymbol{\Sigma}_g^{meso}} + \overline{\boldsymbol{\Sigma}_g^{micro}}\right)$$
$$- \widetilde{\boldsymbol{f}_{gs}} + \overline{\phi_s' \boldsymbol{\nabla}\cdot\boldsymbol{\sigma}_g'} + \rho_g \overline{\phi}_g \mathbf{g}$$

2 phenomena to model:
(i) microscale, and
(ii) mesoscale stress

Graz University of Technology

Common structure for closures (stress)

$$\overline{\boldsymbol{\Sigma}_{\mathrm{q}}^{\mathrm{i}}} = \left[p_{\mathrm{q}}^{\mathrm{i}} - \lambda_{\mathrm{q}}^{\mathrm{i}} \mathrm{tr}\left(\mathbf{D}_{\mathrm{q}}\right)\right]\mathbf{I} - 2\mu_{\mathrm{q}}^{\mathrm{i}} \mathrm{dev}\left(\mathbf{D}_{\mathrm{s}}\right) + \overline{\boldsymbol{\sigma}}_{q,a}^{i}$$

Extended Boussinesq ansatz (as suggested by Cloete [1])

Required:

- Pressure
- Bulk viscosity λ_q (often disregarded)
- Viscosity μ_q
- Residual anisotropic stress tensor contribution $\sigma_{q,a}$ (symmetric tensor)

[1] J.H. Cloete, PhD Thesis, NTNU (2017)

Base structure: unclosed terms in the main solver

Graz University of Technology

Common structure for closures (drag)

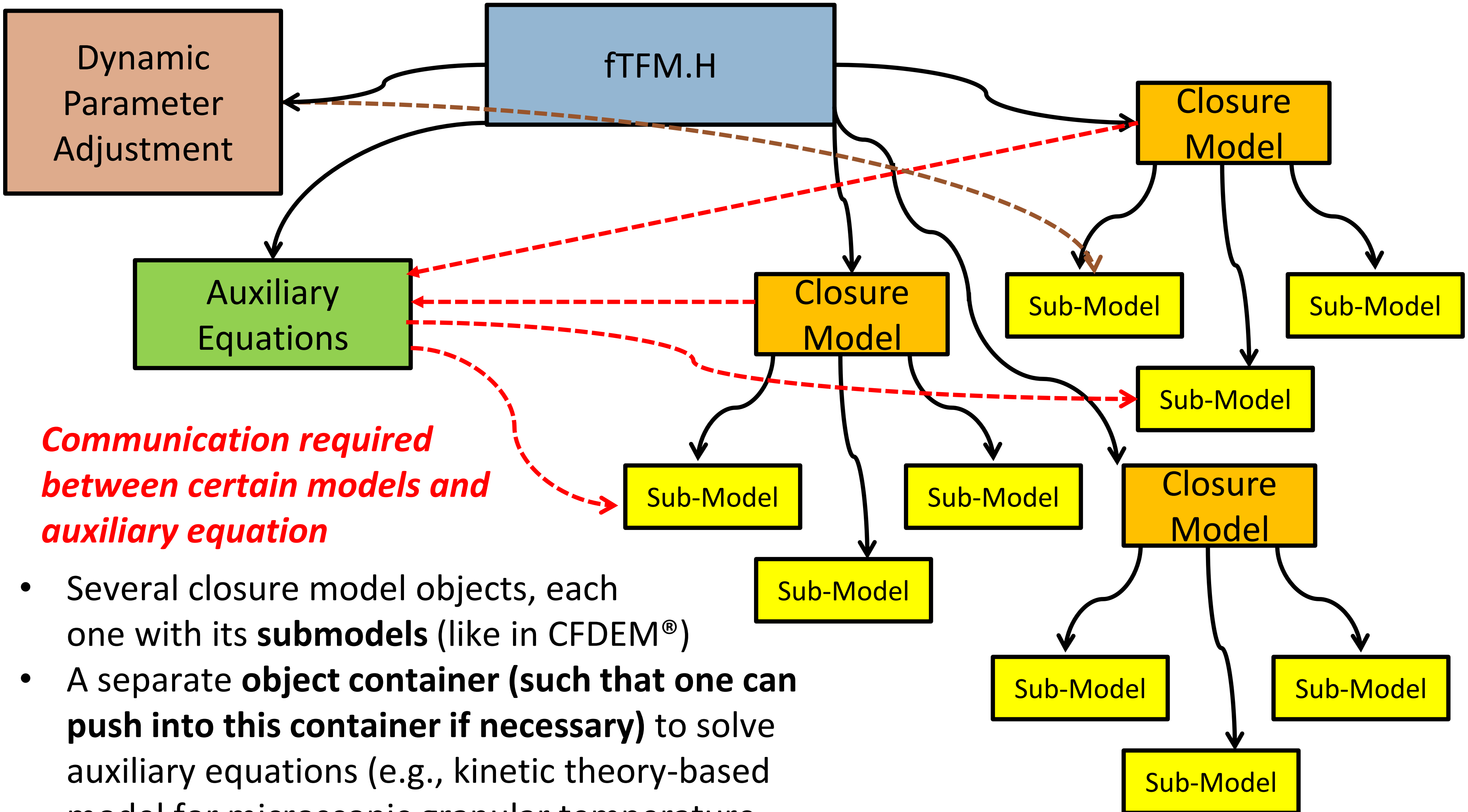
$$\widetilde{\boldsymbol{f}_{\mathrm{gs}}} - \overline{\phi_{\mathrm{s}}' \boldsymbol{\nabla} \cdot \boldsymbol{\sigma}_{\mathrm{g}}'} = \boldsymbol{H_D}\ \widetilde{\beta}\ (\widetilde{\boldsymbol{u}}_{\mathrm{g}} - \widetilde{\boldsymbol{u}}_{\mathrm{s}})$$

Required:

- Closures for microscopic drag coefficient $\widetilde{\beta}$ (available)
- Models for the Heterogeneity factor ***H_D*, should be a tensor.**

Library structure: overall view

TU Graz
Graz University of Technology

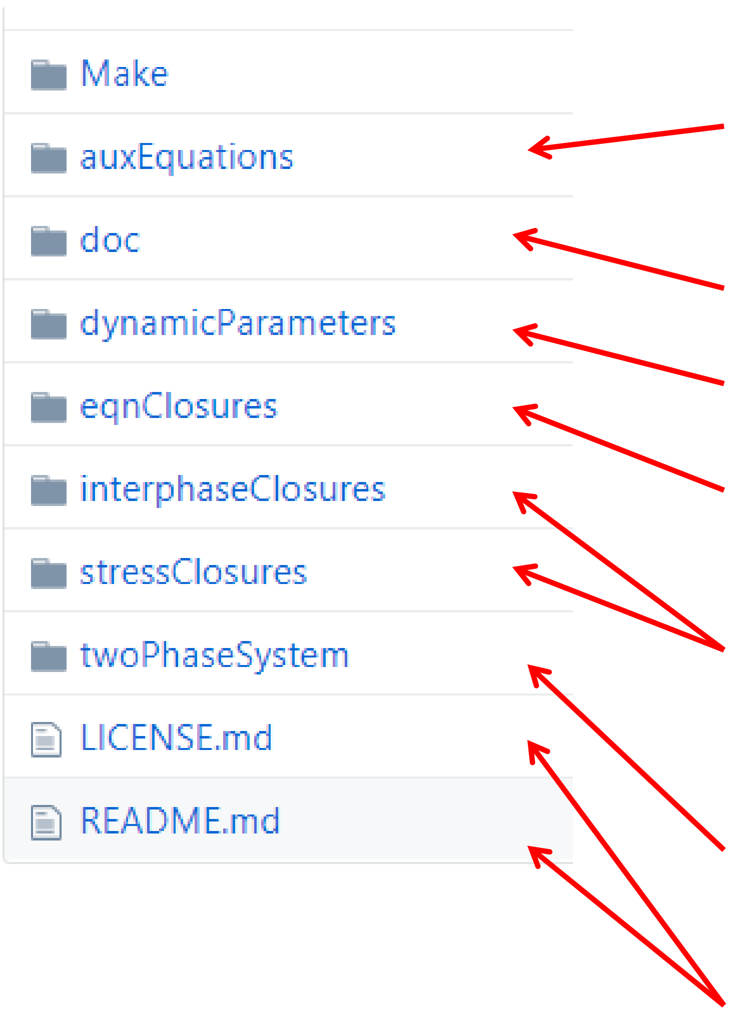


- Several closure model objects, each one with its **submodels** (like in CFDEM®)
- A separate **object container (such that one can push into this container if necessary)** to solve auxiliary equations (e.g., kinetic theory-based model for microscopic granular temperature, filtered granular temperature)

Library structure: overall view 1/4

TU Graz
Graz University of Technology

Folder: /src/eulerian/eulerianFilteredTFM



auxiliary equations to supply closures with additional data (optional)

Markdown-based documentation

Dynamic parameter adjustment (optional)

Additional closures for auxiliary equations

Most important closures models: drag and stress

classes for library organization: these classes hold containers with closures

Instructions for users

Library structure: overall view 2/4

Graz University of Technology

Folder: /src/eulerian/eulerianFilteredTFM/stressClosures/stressSubClosures

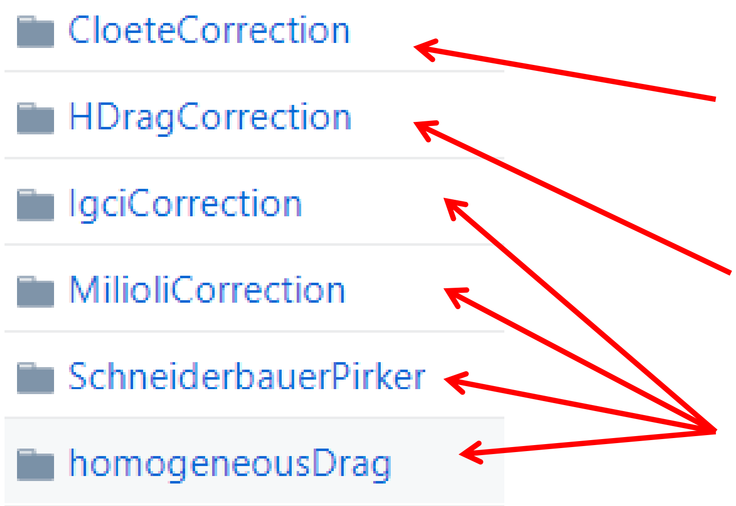
Library structure: overall view 3/4

Folder: /src/eulerian/eulerianFilteredTFM/interphaseClosures/dragClosures/

- HDragCorrections — ***individual closures for drag correction***
- dragClosure — ***base class***
- microDragClosures — ***individual closures for microscale drag closures***

Library structure: overall view 4/4

Folder:
/src/eulerian/eulerianFilteredTFM/interphaseClosures/dragClosures/HDragCorrections/



example closures for drag correction

base class

example closures for drag correction

TU Graz
Graz University of Technology

Generalities about closures and subClosures

- The main closure classes hold a pointer to the main fTFM class (i.e., either ‘twoPhaseSystem’ or ‘pair1In2_’, the phase pair). This is like in CFDEM, where pointers to particleCloud are available
- All main closures classes host containers of their subClosures, and they build the stress(source) by executing subClosures in sequence and summation if needed. This is similar to the CFDEM forceModel and forceSubModels relationship.
- In case closures require additional equations to be solved, then they tell the library to create them. This is done via the equationManager_ object (an autoPtr to an instance of ‘AuxEquations’)
- All sub-closure parameters can be defined in the ‘phaseProperty’ dictionary

Library structure: Closure Model classes

Graz University of Technology

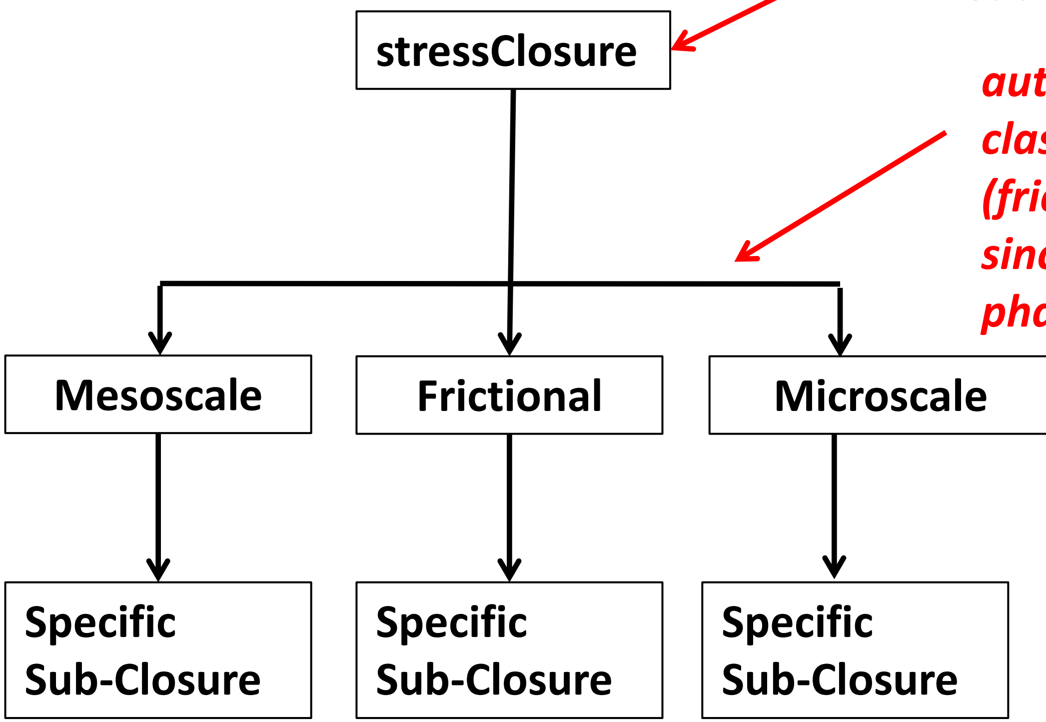
stressClosure class

- Example object: “stressClosure_” (hold by class “phaseModel”)
- A function returns a fvm::fvVectorMatrix object **similar to divDevReff** in turbulence models
- It is in the form $\overline{\boldsymbol{\Sigma}^{\mathrm{i}}_{\mathrm{q}}} = \left[p^{\mathrm{i}}_{\mathrm{q}} - \lambda^{\mathrm{i}}_{\mathrm{q}} \mathrm{tr}\left(\mathbf{D}_{\mathrm{q}}\right)\right] \mathbf{I} - 2\mu^{\mathrm{i}}_{\mathrm{q}} \mathrm{dev}\left(\mathbf{D}_{\mathrm{s}}\right) + \overline{\boldsymbol{\sigma}}^{i}_{q,a}$
- Parameters are provided by submodels (and summed up)
- Submodels **are just calculating the parameters** to save memory and computational time (but sub-model tensors are writeable for debug and monitoring purposes)
- Parameters are calculated **explicitly**, i.e. based on the previous time step
- The stress tensor should be customizable in order to save computational time (for example, disregarding the anisotropic contribution)

Graz University of Technology

stressClosure class

the class that holds stress information (one object for each phase)



autoPtr to 3 instances of class 'StressSubClosure' (frictional can be empty, since non-existing for gas phase)

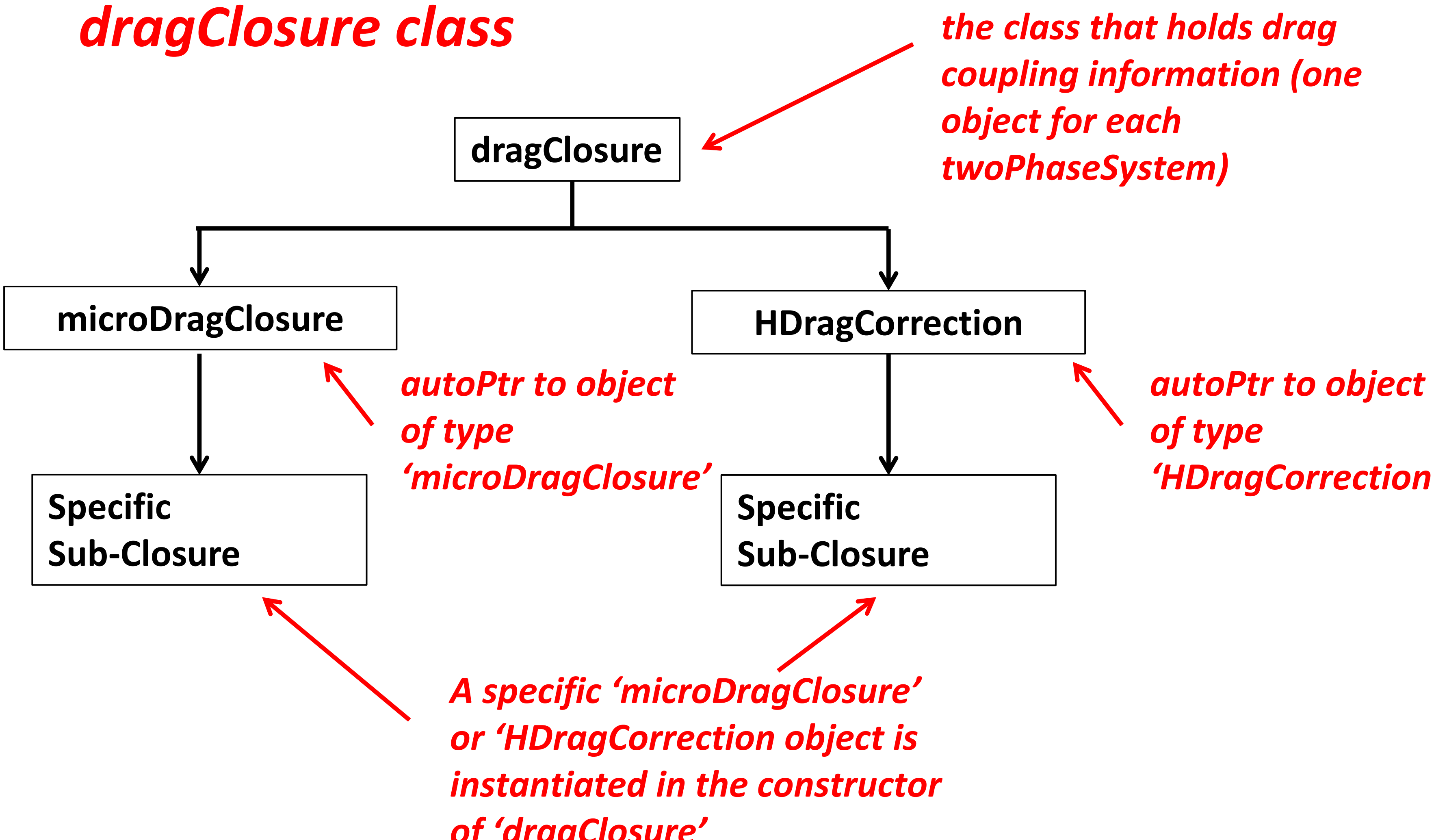
A specific 'StressSubClosure' object is instantiated in the constructor of 'stressClosure'

dragClosure class

- Example object: drag_ (held by class ‘twoPhaseSystem’)
- Does not need to be named **named**.
- A function returns a tensor field
- It always assumes the drag is calculated in the form $\boldsymbol{H_D}\ \widetilde{\beta}\ (\widetilde{\boldsymbol{u}}_{\mathrm{g}} - \widetilde{\boldsymbol{u}}_{\mathrm{s}})$
- Parameters are provided by submodels
- Submodels **are just calculating the parameters** to save memory and computational time (but sub-model quantities should be writeable)
- Parameters are calculated **explicitly**, i.e. based on the previous time step

TU Graz
Graz University of Technology

dragClosure class

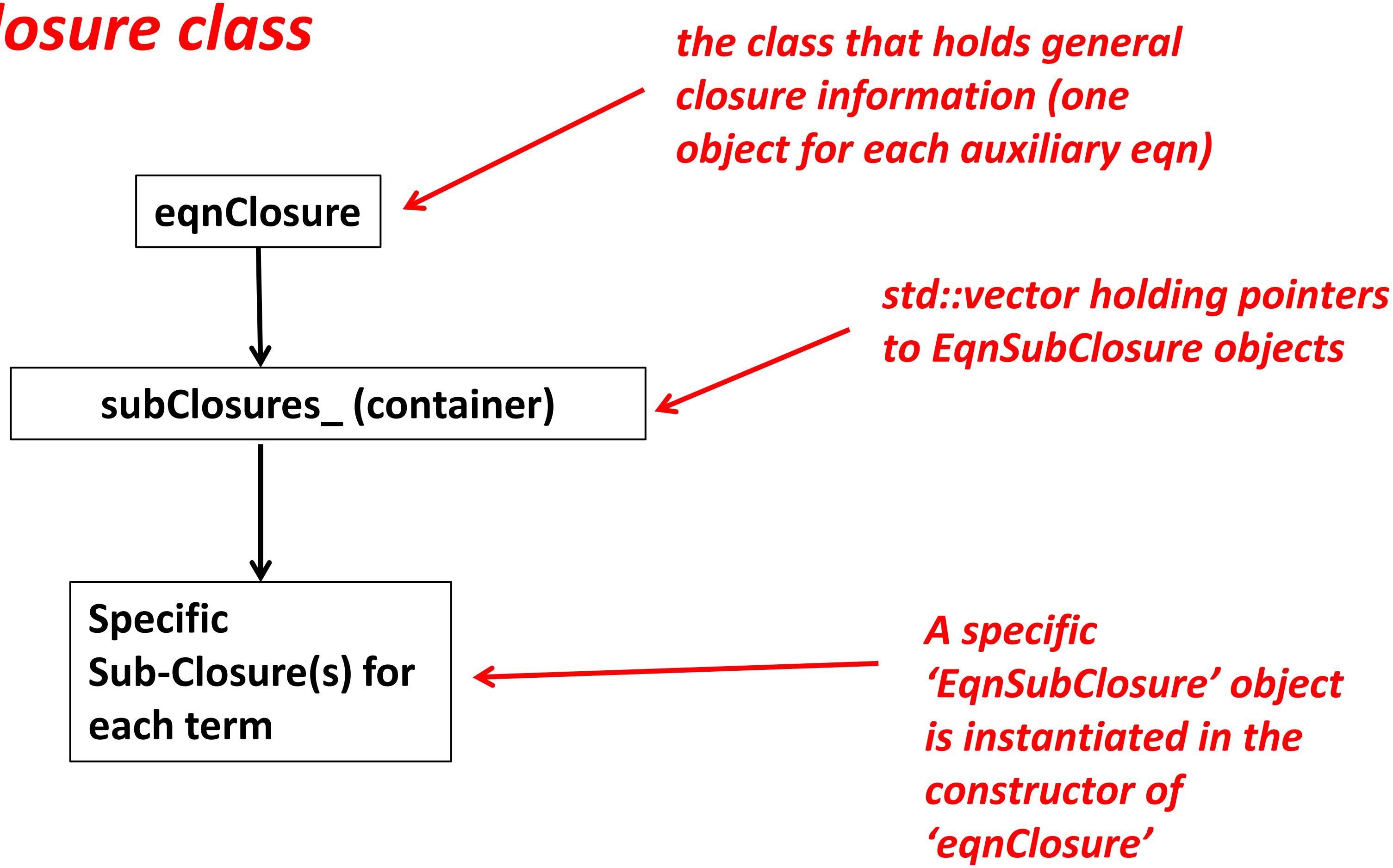


eqnClosure class

- Example object: tmp (will be pushed back into a container in the 'AuxEquations' class)
- It should be **named**. This name is used to lookup submodels in the dictionary
- A function returns a fvm::fvScalarMatrix object **similar to divDevReff** in turbulence models
- It has no specific form
- Parameters are provided by submodels
- Submodels **are just calculating the parameters** to save memory and computational time (but sub-model sources should be writeable)
- Parameters are calculated **explicitly**, i.e. based on the previous time step
- Structure similar to subGridStressModel

Graz University of Technology

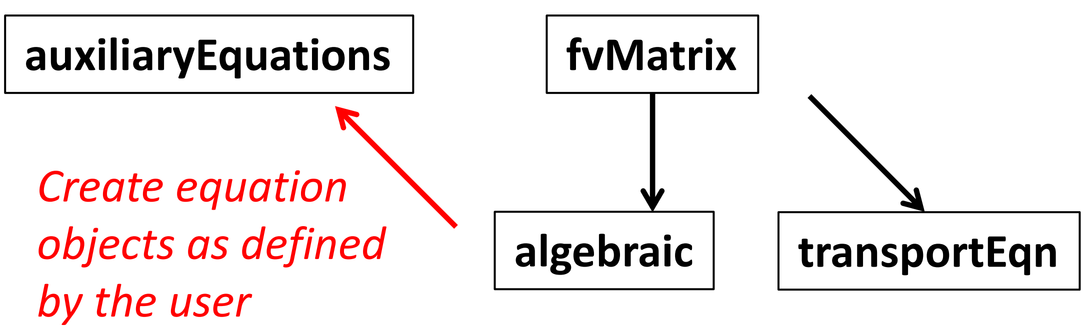
eqnClosure class



Library structure: AuxEquations class

Basic functioning

- Similar to scalarTransportModel, the ‘AuxEquations’ class allows the definition of multiple equations (differential, i.e., ‘transport’, and algebraic equations)
- Transport equations are (for example) in the form: D/Dt (f) = sources, and the method ‘closures[i]->closeEquation’ is used to add closure terms (which are user-defined).
- Therefore, the main class creates and stores the scalars/vector/tensor fields, and holds a the equation information in the struct ‘eqnInfo_’



Library structure: DynamicParameters class

Graz University of Technology

Basic functioning

- It should filter the solid volume fraction to correct the parameters in the drag force
- Still to design, but I will just filter the filtered solid fraction variance to provide better estimation of the solid fraction variance to be used in the calculation of the heterogeneity factor for some drag models.
- It can be automatically triggered by the proper heterogeneity model together with the proper auxiliary equation for the filtered solid fraction variance.