How to extend the functionality of NS-EOF: Example "Turbulence"

November 28, 2023

- **Extend data structure:**
  In case that new degrees of freedom are required (such as local viscosities or additional buffers for velocity values etc.), just inherit from FlowField (let's call the new class TurbulentFlowField) and include the new data fields in this new class. Instead of creating the FlowField in the Main.cpp, you may now just create an object of the TurbulentFlowField. If you compile the code now, everything should still work. You will thus use the old data fields, although the new fields are already available (but are not used yet).

- **Define new functionality:**
  For the default CFD simulation, *Stencil* and *Iterator* implementations are used to define functionality on our data structure. In order to create new functionality that invokes operations on the TurbulentFlowField, you can just create new Stencil class which make use of the TurbulentFlowField: ($\rightarrow$ example: class ComputeLocalViscosityStencil: public FieldStencil <TurbulentFlowField>). Instead of using FlowField in the constructor, just use TurbulentFlowField. Afterwards, this stencil is ready to be applied on the TurbulentFlowField using a FieldIterator <TurbulentFlowField> computeLocalViscosityIterator(turbulentFlowField, computeLocalViscosityStencil).

- **Put functionality together**:
  In order to include new functionality in the algorithm the operations in one time step need to be changed. Therefore, inherit from the class Simulation (we call it TurbulentSimulation in this case) which is initialised by TurbulentFlowField (instead of FlowField). Don't forget to initialise the Simulation class in the TurbulentSimulation constructor ($\rightarrow$: TurbulentSimulation(Parameters &parameters, TurbulentFlowField &turbulentFlowField): Simulation(parameters,turbulentFlowField)). Inside the class TurbulentSimulation, you can define

  – a reference of type TurbulentFlowField which references the new flow field. This yields access to the new data structures.

- stencils and iterators that work on the TurbulentFlowField. You can define them analogously as the FlowField-iterators and stencils are defined in the class Simulation.
- a new implementation of solveTimestep(). Since you will probably need new operations during one time step you can just re-write this method, using the iterators from the class Simulation as well as your new iterators and stencil operations.

Instead of instantiating an object of Simulation in the Main.cpp, you can now instantiate an object of TurbulentSimulation. And whoop—your new algorithm should be carried out now instead of the default one.

That's it. You can just extend NS-EOF by your own needs. Advantages:

- you can re-use all functionality provided so far.
- you do not have to touch anything (like pressure solver etc.)
- you can eventually even re-use parts of the algorithm or at least algorithm phases (such as grid iterations)