Code report

Mingjian Lu

02/12/2016

Contents

1 About files

Original Codes:	This is the original Matlab code, which are the reference codes you should be looking at if you see something really strange or have trouble with the specific calculation steps. They not easy to understand because variable naming is not very intuitive, but will be helpful if you use both python code and Matlab code to understand what we are doing here.
Python Translation:	This is the codes before I did refactoring. The codes have not been OOP-ed, so they are direct translation of Matlab code, so the procedures are nearly identical. Periodic and homogeneous cases in refactoring are copied from here and get modified, I fixed several bugs in these two cases.
Python Refactoring:	As the name suggested, python refactoring is the OOP-ed version of Matlab code, which will be the major part of your work. More will be said on files in this foler later.

1.1 Original Codes

Vanilla MCode	Unmodified Matlab codes, this is original codes
Modified MCode	Clay Shieh, the student was on this project before me, modified matlab code so he can make cross comparison between intermediate products of two functions in Matlab and python, this folder contains his modified code of periodic case
csv (folder)	Folder used to store intermediate outputs
compare.py	Python script for comparing csv files, here used to compare matrices
steady-state.py & .config	A class abstracting inputs for python simulations

1.1.1 Vanilla MCode

New_SN_bench_solver_MC.m	Stochastic case finite step solver, which will create and calculate one randomized grid
New_SN_bench.m	Stochastic case benchmark, which will run its solver a lot of times and average the results
SN_per_bench_solver.m	Periodic case finite step solver, which will calculate one periodic grid
periodic_benchmark.m	Periodic case benchmark, which will run its solver a lot of times and average the results
SN_hom.m	Because homogeneous case is relatively easier, we didn't separate its benchmark and solver

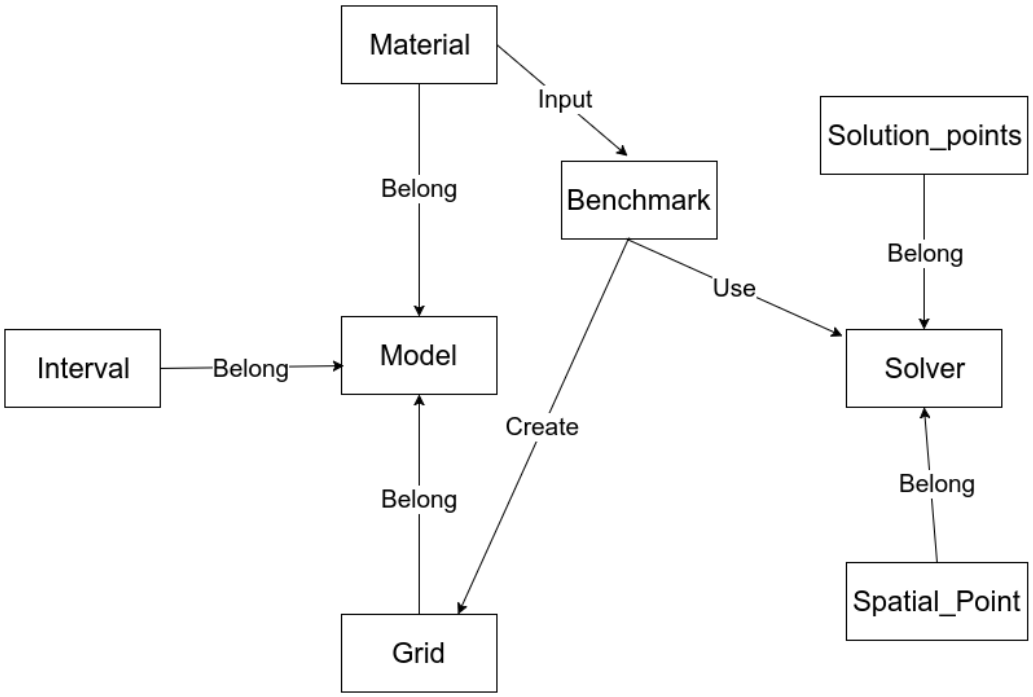
1.2 Python Translation

compare.py	Used to compare two csv files
homogeneous.py	Homogeneous case benchmark
nonhomogeneous.py	Stochastic finite step benchmark and its solver
periodic.py	Periodic case benchmark and its solver
steady-state.py	Input wrapper and config files parser

1.3 Python Refactoring

Config.py	Input wrapper and parser, I didn't use this functionality
OneDGridModel.py	Model classes, used to generate different grids for three cases, namely, homogenuous, periodic and stichastic
OneDGridModelBenchMark.py	Benchmark classes, which functions as API exposed to users, using model and solver to simulate three cases
OneDGridModelSolver.py	Solver classes, which solve the result for one certain grid
SolverTest.py	Test script, also you can see this as the supposed way to use benchmarks

2 Abstractions of Python Refactoring



These are primary components involved in three main classes: Benchmark, Solver and Model.

Model takes care of building grids and materials, using intervals to represent an occurrence of a material, and Grid to represent distribution of materials in a specific grid.

Solvers are for matrix constructing and solving, since we need to discretize the grid (add more points other than interfaces), we need to use Spatial_points to keep our discretization. Also since not all our calculated points are needed to re-interpolate the final answer, we need a Solution_point class to keep track of which points are needed and to integrate information.

Benchmark is the one to utilize model and solver to solve a proposed question.

3 Known BUG

Model_1D_Stochastic_Finite_Step_Solver interpolate the left second point and second right points in a unstable way, probably has something to do with solve_required_points or solve_scalar_flux.

4 Docs for classes

I comment this part in code.