`Geophysical Fluids Modeling Framework` Handbook

`GAME` Development Team

August 13, 2021

All physical quantities in this document are to be multiplied with their respective SI units.

Contents

Chapter 1

Overview

This is only the handbook (manual) of the `Geophysical Fluids Modeling Framework (GAME)`, it explains how to configure, compile and run (use) the model. For a scientific derivation of the model see the documentation and the literature cited therein. The source code of the project is maintained on github (`https://github.com/OpenNWP/GAME`), this is also the place to ask questions or report errors. It is never wrong to think a bit before you post something there.

The `GAME` project incorporates four different executables:

- `grid_generator`, a program for creating model grids
- `orography_generator`, a tool for creating orography files
- `test_generator`, a tool for creating initialization states of test scenarios
- `game`, the model executable itself

Chapter 2

Installation

2.1 Dependencies

The following dependencies must be installed before being able to successfully build the model:

- geos95 (`https://github.com/MHBalsmeier/geos95`)
- netcdf library (Ubuntu: `sudo apt-get libnetcdf-dev`)
- eccodes library (installation manual: `https://mhbalsmeier.github.io/tutorials/eccodes_on_ubuntu.html`)
- CMake (Ubuntu: `sudo apt-get install cmake`)
- atmostracers (`https://github.com/MHBalsmeier/atmostracers`)
- OpenMPI (Ubuntu: `sudo apt-get install mpich`)

2.1.1 For using the plotting routines

The following packages are additionally required if you want to make use of the plotting routines:

- Python and the visualization library scitools-iris (installation manual: `https://mhbalsmeier.github.io/tutorials/iris_on_ubuntu.html`, only for the plotting routines, which are of course not art of the actual model)
- FFMPEG (Ubuntu: `sudo apt-get install ffmpeg`, only for the plotting routines)

2.1.2 For developing

- Valgrind (Ubuntu: `sudo apt-get install valgrind`, for doing checks)

2.2 Building

`CMake` is used for building `GAME`. The building process is managed using the bash scripts in the directory `build_scripts`. The following list gives an overview of the scripts residing in this directory:

- `build_install.sh`: The installation directory is controlled by the variable `aim_dir`.
- `install_grids.sh`: Installs the grids to the installation directory.
- `install_tests.sh`: Installs the test state initialization files to the installation directory.
- `install_run_scripts.sh`: Installs the run scripts to the installation directory.
- `install_plotting_routines.sh`: Installs the plotting routines to the installation directory.
- `install_everything.sh`: Executes all the other install scripts.

Scripts with the suffix `_dev` are not different from the original scripts, they only allow choosing a different installation directory for installations of test versions.

Chapter 3

Running the model

The configuration of the model must be set in two different files:

- `core/src/enum_and_typedefs.h`: modify `RES_ID`, `NO_OF_LAYERS`, `NO_OF_GASEOUS_CONSTITUENTS` and `NO_OF_CONDENSED_CONSTI` These must conform with the grid file and the initialization state file. It must be done before the compilation.
- The run script: one of the files contained in the directory `run_scripts`. The comments in these files explain the meaning of the variables. This can be done after the compilation.

Since the files `core/src/enum_and_typedefs.h` and `core/src/settings.c` are part of the model's source code, the model must be recompiled if something is changed in them. Alternatively, one can compile several executables and name them according to their configuration.

3.1 Dynamics configuration

3.2 Physics configuration

3.2.1 Local thermodynamic equilibrium option

Assuming a local thermodynamic equilibrium in a heterogeneous fluid boils down to assuming that all constituents have the same temperature. This reduces the complexity of the simulation by about 40 %, since now internal energy densities are not prognostic variables anymore.

Chapter 4

Grid generation

4.1 Horizontal grid properties

The horizontal grid structure is determined by the following properties:

- the resolution, specified via the parameter `RES_ID`
- the optimization

4.2 Vertical grid properties

The vertical grid structure is determined by the following properties:

- the height of the top of the atmosphere, specified via the parameter `TOA`
- the number of layers, specified via the parameter `NUMBER_OF_LAYERS` N_L
- the number of layers following the orography, specified via the parameter `NUMBER_OF_ORO_LAYERS` N_O
- the stretching parameter β, which can be set in the run script
- the orography, specified via the parameter `ORO_ID`

The generation of the vertical position of the grid points works in three steps:

1. First of all, vertical positions of preliminary levels with index $0 \leq j \leq N_L$ are determined by

$$z_j = T\sigma_{z,j} + B_j z_S, \tag{4.1}$$

where T is the top of the atmosphere, $\sigma_{z,j}$ is defined by

$$\sigma_{z,j} := \left(1 - \frac{j}{N_L}\right)^{\alpha}, \tag{4.2}$$

where $\alpha \geq 1$ is the so-called *stretching parameter*, z_s is the surface height and B_j is defined by

$$B_j := \frac{j - (N_L - N_O)}{N_O}. \tag{4.3}$$

2. Then, the scalar points are positioned in the middle between the adjacent preliminary levels.
3. Then, the vertical vector points are regenerated by placing them in the middle between the two adjacent layers.
4. Finally, the vertical positions of the other points are diagnozed through interpolation.

4.3 How to generate a grid

The grid generator needs to be recompiled for every specific resolution, top height, number of layers as well as number of orography following layers. Therefore change the respective constants in the file `grid_generator.c` and execute the bash script `compile.sh`. Then run the grid generator using the bash script `run.sh` with the desired `ORO_ID`. Table 4.1 explains all the parameters to be set in `run.sh`. Optimized grids have the postfix `_SCVT`.

name	domain	meaning
`ORO_ID`	all value for which an orography is defined	orography ID
`optimize`	0, 1	optimization switch (fails if `ORO_ID` is not 0)
`n_iterations`	integer ≥ 1	number of iterations (ignored if `optimize` = 0), 2000 seems to be a safe value
`use_scalar_h_coords_file`	0, 1	switch to determine wether horizontal coordinates of triangle vertices (generators of the grid) shall be used from another file
`scalar_h_coords_file`	string	input file for dual triangle vertices (only relevant if `use_scalar_h_coords_file` = 1)
`stretching_parameter`	≥ 1, real	defines the vertical stretching of the grid, one means no stretching

Table 4.1: Grid generator run script explanation.

Chapter 5

Generating a new orography file

Orography files are generated with the code residing in the directory `orography_generator/src`. Firstly, change the parameter `RES_ID` in the file `orography_generator.c` to the desired value and compile. Then source the bash scribt `run.sh` with the desired `ORO_ID`. Tab. 5.1 shows the definition of the orography IDs. Real orography can be downloaded from

- `https://psl.noaa.gov/cgi-bin/db_search/DBSearch.pl?Dataset=NCEP+Reanalysis &Variable=Geopotential+height&group=0&submit=Search` (`ORO_ID` = 3)

These files are stored in the directory `orography_generator/real`. An information file explains them and defines their individual `ORO_ID`s. A $1/r$-interpolation with four values is used to interpolate the data to the scalar data points.

`ORO_ID`	**Description**
0	no orography
1	Gaussian mountain of 8 km height and 224 m standard deviation located ad 0 N / 0 E
≥ 2	real orography

Table 5.1: Definition of orography IDs.

Chapter 6

Generating a new test state file

A new test state can be generated with the code in the directory `test_generator/src`. Therefore, firstly change the parameters `RES_ID`, `NUMBER_OF_LAYERS` and `NUMBER_OF_ORO_LAYERS` in the file `test_generator.c`. Then compile by sourcing the file `compile.sh` before executing the file `run.sh` with the specific `test_id`. Tab. 6.1 shows the definition of the test IDs.

`TEST_ID`	**Description**
0	standard atmosphere without orography (`ORO_ID` = 0)
1	standard atmosphere with Gaussian mountain (`ORO_ID` = 1)
2	standard atmosphere with real orography (`ORO_ID` = 2)
3	dry Ullrich test with `ORO_ID` = 0
4	dry Ullrich test with `ORO_ID` = 1
5	dry Ullrich test with `ORO_ID` = 2
6	moist Ullrich test with `ORO_ID` = 0
7	moist Ullrich test with `ORO_ID` = 1
8	moist Ullrich test with `ORO_ID` = 2

Table 6.1: Definition of test IDs.