Geophysical Fluids Modeling Framework Handbook

GAME Development Team

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

Contents

Chapter 1

Overview

The Geophysical Fluids Modeling Framework (GAME) is a non-hydrostatic hexagonal C grid dynamical core with the possibility to advect a variable number of constituents. The term *dynamical core* typically refers to the simulation of a dry atmosphere. Everything else is then referred to as *physics*. Diffusive terms, including turbulence parameterizations, are sometimes understood to be part of the dynamical core and sometimes seen as part of the model's physics. The dry air is in this understanding a "carrier medium", whereas constituents, including water in different phases, are usually only passively advected. This thinking always leads to deep physical inconsistencies during later stages of model development, whose impact on forecast and climate simulation accuracy remains unknown.

Therefore, a new, capaple framework for simulating geophysical fluids is necessary and, due to the advent of even more powerful computers, also realistic. The GAME can be seen as a dynamical core, but not in the traditional sense, instead rather in a modernized sense. Its aim is to simulate the dynamics of geophysical fluid flow accurately, and at the same time make it possible to couple the model to different constituents *consistently*, which means without violating fundamental physical constraints. Following a modular way of thinking, the actual quantifications of the constituent's source terms have been split off into the atmostracers library [2], which is meant to evolve into a wide collection of source terms for atmospheric constituents, also including chemistry and parameterizations. For radiation, it is coupled to the RTE+RRTMGP (Radiative Transfer for Energetics + Rapid and Accurate Radiative Transfer Model for Geophysical Circulation Model Applications-Parallel) [8], [10] scheme, which follows a similar approach to radiation simulation as GAME follows to fluid simulation.

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 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
- surface_generator, a tool for creating surface properties
- test_generator, a tool for creating initialization states of test scenarios
- game, the model executable itself

Chapter 2

Code structure

The code of the model resides in the directory `src`.

2.1 Spatial operators

- Coriolis: [1] and [9] modified by [5]
- kinetic energy: [4]

2.2 Time stepping

A fully Eulerian time stepping is employed. The basic building structure is Runge-Kutta second order (RK2). In the vertical, at every substep, an implicit column solver is used, which makes it possible to violate the CFL criterion of vertically propagating sound and fast gravity waves. This has the cost of decreasing the accuracy of these modes, which is however a bearable trade-off, since these waves are of low meteorological relevance. Furthermore, a forward-backward scheme is used, where the divergence term is backward.

Chapter 3

Installation

3.1 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`)

3.1.1 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)

3.1.2 For developing

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

3.2 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 4

Grid generation

4.1 Fundamental grid quantities

The following quantities are sufficient to uniquely define the grid:

- the horizontal coordinates of the generating points
- the numbering of the generating points
- the numbering and orientation of the horizontal vectors
- the numbering of the dual cell mid points
- The orientation of the dual horizontal vectors. Since their horizontal positions coincide with the horizontal positions of the primal vectors, both sets of vectors can be numbered in the same way.

All other quantities, be it floating point numbers like grid box volumes or areas, or integer quantities like neighborhood relationships, can be implicitly derived. Consequently, once must firstly focus on the fundamental grid properties.

4.1.1 Creating a spherical geodesic grid

By *horizontal grid* we mean the grid on one single layer. Without loss of generality this layer can be assumed to coincide with the surface of the unity sphere.

4.1.2 Numbering and orienting the vector points

4.1.3 Numbering and orienting the dual vector points

Until now, we have only operated with the six following arrays:

- `latitude_scalar`, `longitude_scalar`
- `from_index`, `to_index`
- `from_index_dual`, `to_index_dual`

4.2 Derived quantities

4.2.1 Horizontal coordinates of dual scalar points

The dual cells of a hexagonal grid are the triangular grid cells. Since we aim at an orthogonal grid, the dual scalar points must be the Voronoi centers of the triangular cells. Label the vertices of one triangle with $\mathbf{r}_0, \mathbf{r}_1, \mathbf{r}_2$, then its Voronoi center $\mathbf{r}_v$ is located at the position

$$\mathbf{r}_v := \frac{(\mathbf{r}_1 - \mathbf{r}_0) \times (\mathbf{r}_2 - \mathbf{r}_0)}{|(\mathbf{r}_1 - \mathbf{r}_0) \times (\mathbf{r}_2 - \mathbf{r}_0)|}. \tag{4.1}$$

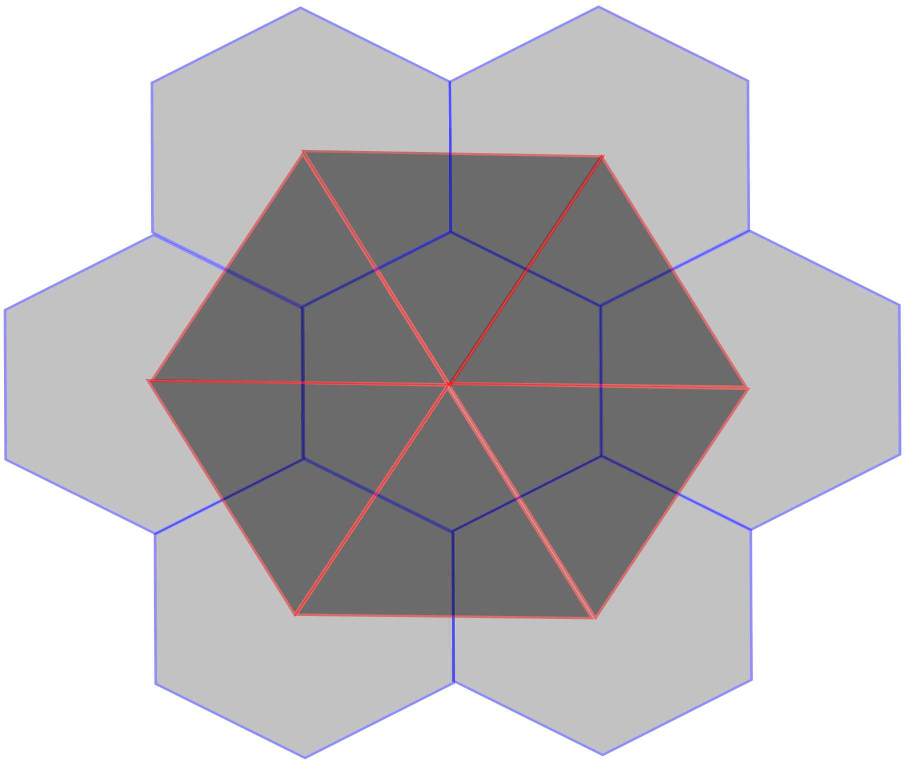

Figure 4.1: A subset of a regular horizontal hexagonal grid. The hexagonal grid (blue lines) and the triangular grid (red lines) form a pair of a primal-dual grid. In `GAME`, the hexagonal grid is the primal grid, while the triangulars form the dual one. In a triangular grid model it is the other way around. During the grid generation procedure we refer to the triangle edge points (hexagon centers) as the generating points or generators, for short.

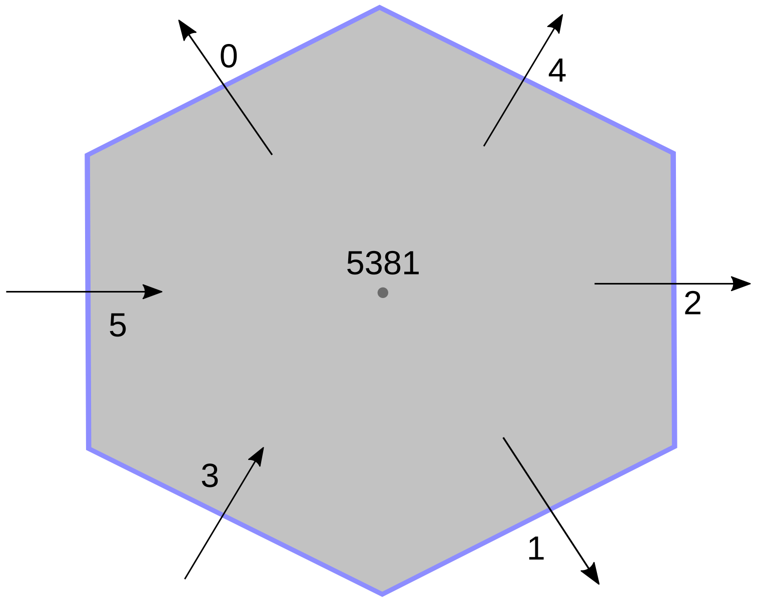

Figure 4.2: A sample hexagon with a horizontal scalar index of 5381. The directions of the arrows indicate the directions of unit vectors at cell edges. The drawn orientations would lead to `adjacent_vector_signs_h`$[6\cdot 5381+0]=1$, `adjacent_vector_signs_h`$[6\cdot 5381+1]=1$, `adjacent_vector_signs_h`$[6\cdot 5381+2]=1$, `adjacent_vector_signs_h`$[6\cdot 5381+3]=-1$, `adjacent_vector_signs_h`$[6\cdot 5381+4]=1$, `adjacent_vector_signs_h`$[6\cdot 5381+5]=-1$.

4.2.2 Finding the neighboring vector points of a primal cell and their orientation

4.2.3 Finding the neighboring vector points of a dual cell and their orientation

4.3 Grid optimization

Hexagonal spherical grids need to be optimized for numerical modeling. Therefore, the Lloyd algorithm is used, which yields a *spherical centroidal Voronoi tesselation (SCVT)* after convergence [3]. [7] gives an overview of optimization alternatives and it seems to be that the SCVT is the most suitable for modeling. The procedure employed for executing the Lloyd algorithm is the one described in [6].

4.4 Vertical grid structure

So far, only a horizontal grid has been examined. The grid generator, however, shall produce full three-dimensional grids. In order to simplify matters, the following conventions are made:

- Since the vertically oriented primal vector points have the same horizontal coordinates as the primal scalar points, their horizontal numbering is also the same.
- Since the vertically oriented dual vector points have the same horizontal coordinates as the dual scalar points, their horizontal numbering is also the same.

4.5 Scalability

The computation time of the most expensive for loops scale with N^2, where N is the number of horizontal grid points. This means that doubling the horizontal resolution (four times as much horizontal grid points) leads to a 16 times longer computation time of the grid generator. This is similar to the model itself, where a doubling of the horizontal and vertical resolution and a halfening of the time step leads to 16 times longer integration times. Therefore, the largely implicit formulation of the grid generator posos no problem to its performance at higher resoultions.

4.5.1 Permutations of the grid points

4.6 Horizontal grid properties

The horizontal grid structure is determined by the following properties:

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

4.7 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 $0 \le j \le N_L$ are determined by

$$z_j = T\sigma_{z,j} + B_j z_S, \tag{4.2}$$

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

$$\sigma_{z,j} := \left(1 - \frac{j}{N_L}\right)^{\alpha}, \tag{4.3}$$

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

$$B_j := \frac{j - (N_L - N_O)}{N_O}. \tag{4.4}$$

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.

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.

4.8 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`.

Chapter 5

Generating a new surface file

The surface generator resides in the directory `surface_generator` and is an individual executable meant to produce surface properties (orography, roughness length, radiative properties, etc.) for different cases defined by the `oro_id`, given the horizontal positions of the primal scalar points. Execute the script `run._script.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 `surface_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.

The so-called *full width at half maximum (FWHM)* is the width of a peak at half its maximum height. For a normal distribution, one finds

$$\begin{aligned}\exp\left(-\frac{x_0^2}{2\sigma^2}\right) = \frac{1}{2} = 2^{-1} \Leftrightarrow -\frac{x_0^2}{2\sigma^2} = -\ln(2)\\ \Leftrightarrow x_0^2 = 2\sigma^2\ln(2) \Leftrightarrow x_0 = \sigma\sqrt{2\ln(2)}.\end{aligned} \tag{6.1}$$

This implies

$$\text{FWHM} = 2x_0 = \sigma\sqrt{8\ln(2)} \Rightarrow \sigma = \frac{\text{FWHM}}{\sqrt{8\ln(2)}}. \tag{6.2}$$

`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.

Chapter 7

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.

7.1 Dynamics configuration

7.2 Physics configuration

7.2.1 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.

7.3 Coupling to the radiation field

`GAME` employs the so-called `RTE+RRTMGP (Radiative Transfer for Energetics + Rapid and Accurate Radiative Transfer Model for Geophysical Circulation Model Applications-Parallel)` [8], [10] scheme.

Chapter 8

Configuring output

Bibliography

[1] J. Thuburn et al. Numerical representation of geostrophic modes on arbitrarily structured C-grids. In: *Journal of Computational Physics* 228 (22 2009), pp. 8321–8335.

[2] *Atmosconstituents github repository*. Oct. 22, 2020. URL: https://github.com/MHBalsmeier/atmostracers.

[3] Qiang Du, Max D. Gunzburger, and Lili Ju. Constrained Centroidal Voronoi Tesselations for Surfaces. In: *SIAM J. Sci. Comput.* 24.5 (Apr. 2003), pp. 1488–1506.

[4] Almut Gassmann. A global hexagonal C-grid non-hydrostatic dynamical core (ICON-IAP) designed for energetic consistency. In: *Quarterly Journal of the Royal Meteorological Society* 139.670 (2013), pp. 152–175. DOI: 10.1002/qj.1960. eprint: https://rmets.onlinelibrary.wiley.com/doi/pdf/10.1002/qj.1960. URL: https://rmets.onlinelibrary.wiley.com/doi/abs/10.1002/qj.1960.

[5] Almut Gassmann. Discretization of generalized Coriolis and friction terms on the deformed hexagonal C-grid. In: *Quarterly Journal of the Royal Meteorological Society* 144.716 (2018), pp. 2038–2053. DOI: 10.1002/qj.3294. eprint: https://rmets.onlinelibrary.wiley.com/doi/pdf/10.1002/qj.3294. URL: https://rmets.onlinelibrary.wiley.com/doi/abs/10.1002/qj.3294.

[6] Hiroaki Miura and Masahide Kimoto. A Comparison of Grid Quality of Optimized Spherical Hexagonal–Pentagonal Geodesic Grids. In: *Monthly Weather Review* 133.10 (Oct. 2005), pp. 2817–2833. ISSN: 0027-0644. DOI: 10.1175/MWR2991.1. eprint: https://journals.ametsoc.org/mwr/article-pdf/133/10/2817/4213997/mwr2991_1.pdf. URL: https://doi.org/10.1175/MWR2991.1.

[7] Pedro S. Peixoto and Saulo R.M. Barros. Analysis of grid imprinting on geodesic spherical icosahedral grids. In: *Journal of Computational Physics* 237 (2013), pp. 61–78. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2012.11.041. URL: http://www.sciencedirect.com/science/article/pii/S0021999112007218.

[8] Robert Pincus, Eli J. Mlawer, and Jennifer S. Delamere. Balancing Accuracy, Efficiency, and Flexibility in Radiation Calculations for Dynamical Models. In: *Journal of Advances in Modeling Earth Systems* 11.10 (2019), pp. 3074–3089. DOI: 10.1029/2019MS001621. eprint: https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1029/2019MS001621. URL: https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1029/2019MS001621.

[9] Todd Ringler et al. A unified approach to energy conservation and potential vorticity dynamics on arbitrarily structured C-grids. In: *J. Comput. Physics* 229 (May 2010), pp. 3065–3090. DOI: 10.1016/j.jcp.2009.12.007.

[10] *RTE-RRTMGP github repository*. June 22, 2020. URL: https://github.com/earth-system-radiation/rte-rrtmgp.