GAME Handbook

GAME Development Team

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

Chapter 1

Overview

The GAME projects 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
- eccodes library (installation manual: `https://mhbalsmeier.github.io/tutorials/eccodes_on_ubuntu.html`)
- CMake
- atmostracers (`https://github.com/MHBalsmeier/atmostracers`)
- rte-rrtmgp-c (`https://github.com/MHBalsmeier/rte-rrtmgp-c`)
- Python (only for the plotting routines, which are of course not part of the actual model)
- OpenMPI (not yet)

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:

- `debug.sh`:
- `build.sh`: The installation directory is controlled by the variable `aim_dir`.
- `install_output.sh`:
- `build_install.sh`:
- `install_plotting_routines.sh`:
- `install_run_scripts.sh`:

Scripts with the suffix `_dev` are not different from the original scripts, they 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 three different files:

- `src/enum_and_typedefs.h`: modify `RES_ID`, `NO_OF_LAYERS` and `NO_OF_ORO_LAYERS`. These must conform with the grid file and the initialization state file.
- The file `src/settings.c`: modify the characteristics of damping at the model top and diffusion.
- The run script: explanation below.

Listing 3.1: Example input file.

```bash
#!/bin/bash
operator=MHB
overwrite_run_id=1
run_id=jw_pert_moist
run_span=43200
write_out_interval=900
grid_props_file=grids/B6L26T30000_O2_OL17_SCVT.nc
init_state_filename=test_5_B6L26T30000_O2_OL17_SCVT.nc
init_state_file=input/$init_state_filename
output_dir_base=output
cfl_margin=0.0
temperature_diff_h=1
temperature_diff_v=1
momentum_diff_h=1
momentum_diff_v=1
tracers_on=1
rad_on=0
radiation_delta_t=3600
write_out_mass_dry_integral=1
write_out_entropy_gas_integral=1
write_out_linearized_entropy_gas_integral=1
write_out_energy_integral=1
# relevant only for OMP
export OMP_NUM_THREADS=5
# relevant only for MPI
number_of_cpus=1
year=2000
month=1
day=1
hour=0
nwp_mode=0
# necessary only for data assimilation
ndvar_directory=/home/max/compiled/ndvar
source core/run.sh
```

Listing 3 is an example of an input file. Table 3.1 explains the meanings of the variables.

name	domain	meaning
operator	string	Operator of the model, for example `Company XYZ, Inc.`
overwrite_run_id	0, 1	if 0: use auto-generated run_id, if 1: use manually set run_id (see next line)
run_id	string (optional)	run_id to be used if overwrite_run_id is set to 1
run_span	integer	How long the model shall run into the future.
write_out_interval	integer ≥ 900	Every how many seconds autput shall be generated.
grid_props_file	string	File name of the grid properties file.
init_state_filename	string	File name of the initialization state file.
init_state_file	string	Full path of the initialization state file.
output_dir_base	string	The directory to which output shall be written.
cfl_margin	double	Manual reduction of the time step below the CFL criterion: $\Delta t = (1 - \text{cfl\_margin})\,\Delta t^{(\text{CFL})}$.
temperature_diff_h	0, 1	horizontal temperature diffusion switch
temperature_diff_v	0, 1	vertical temperature diffusion switch
momentum_diff_h	0, 1	horizontal momentum diffusion switch
momentum_diff_v	0, 1	vertical momentum diffusion switch
dissipation_on	10, 1	dissipation switch
tracers_on	0, 1	tracers switch
rad_on	0, 1	radiation switch
radiation_delta_t	double ≥ Δt	Every how many seconds the radiation flux densities shall be updated.
write_out_mass_dry_integral	0, 1	Switch to decide wether a global integral of dry mass shall be written out at every time step.
write_out_entropy_gas_integral	0, 1	Switch to decide wether a global integral of the entropy shall be written out at every time step.
write_out_energy_integral	0, 1	Switch to decide wether a global integral of the energy shall be written out at every time step.

Table 3.1: Input file explanation.

Chapter 4

Grid generation procedure

A grid is determined by the following five properties:

- the resolution, specified via the parameter `RES_ID`
- the orography, specified via the parameter `ORO_ID`
- the height of the top of the atmosphere, specified via the parameter `TOA`
- the number of layers, specified via the parameter `NUMBER_OF_LAYERS`
- the number of layers following the orography, specified via the parameter `NUMBER_OF_ORO_LAYERS`

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`. Otimized 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)

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	orography of JW test
2	Gaussian mountain of 8 km height and 224 m standard deviation located ad 0 N / 0 E
≥ 3	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
1	standard atmosphere with Gaussian mountain (`ORO_ID` = 2)
2	JW dry unperturbed
3	JW dry perturbed
4	JW moist unperturbed
5	JW moist perturbed
6	JW dry, balanced, with `ORO_ID` = 3
7	JW moist, balanced, with `ORO_ID` = 3

Table 6.1: Definition of test IDs.