HIM Quickstart Guide

William Cooke <William.Cooke@noaa.gov>

Brief instructions for running HIM experiments. Most of the specific documentation for using HIM is contained within the comments in the source code.

Table of Contents

1. Get the source code and scripts through CVS checkout or download

After you do CVS checkout successfully, a directory will be created. For convenience, this directory will be referred to as the `ROOT` directory. A readme file in the `ROOT` directory will tell you the contents of each subdirectory under `ROOT`.

In the sections below, `test_case` is a generic name referring to the name of a specific experiment you are working on. Thus, `test_case` in the path `ROOT/src/HIM_examples/test_case` should be replaced by a concrete name (for example `him2gyre`, `himDOME` , etc.)

2. Input data

There are two ways to specify input data for HIM experiments. They can be coded internally or read from files. For experiments with analytic functions describing the initial conditions, forcing, and topography, these expressions can be directly entered in appropriate places in the HIM code. Input data that is already on the model grid can be read in directly from files. For a given experiment, there can be any mix of variables taken from internal expressions and external files. The examples in `src/HIM_examples/test_case` illustrate both ways the input data can be specified. Comparing corresponding files between examples is a good way to get a quick idea of how this is done.

2.1. Internal specification of data

Fortran (derived originally from "Formula Translation") is a perfectly good language for evaluating analytic algorithms. Within the HIM code, several subroutines are called out as places where users are likely to want to insert their own analytic specifications for initial conditions and surface forcing.

Initial conditions should be specified within `HIM_initialization.F90`. This file contains numerous subroutines, each named to indicate the variable it specifies.

Similarly, surface forcing fields for ocean-only models can be specified by modifying `HIM_surface_forcing.F90`.

It is recommended that users create their own directories, akin to those in `src/HIM_examples/...`, to contain those files that they modify relative to the canonical versions in `src/HIM/...`, and edit the `path_names` file to reflect these new file locations.

2.2. Get data files via NOMADS server

Note
Note that external datasets are only needed for him_global and him_sis.

Users first need to create directory `data` for each test case:

```
mkdir ROOT/data/test_case
```

The data files in NOMADS are organized in directories having the same name as name of `test_case` (data for `him_global` are in directory `data/him_global`, etc.). Go to the experiment of your interest (see Section 6, “Brief description of HIM experiments”below) and start downloading file `test_case.tar.gz` to `ROOT/data/test_case`. You will need to uncompress the file `test_case.tar.gz` after downloading.

Download the sample output tar file. These data are **NOT** needed for running experiments. They are for the purpose of comparing your test-case results with results produced at GFDL.

3. Parameter and option specification

HIM has a number of options and parameters that the user can select, as appropriate for a particular configuration. These choices are made at compile time by editing the include file `HIM_init.h`, but many of these can be overridden at run time by parsing a file with an identical syntax to `HIM_init.h`. This run-time file is specified as the HIM_namelist variable `parameter_filename`, HIM_input in these test cases. `HIM_init.h` has extensive comments that document the meaning and units of each such option or parameter.

The parameters that can not be specified at run time are primarily those that the compiler needs to know about to allocate memory statically - namely the total domain size, the layout of the processors, and which coordinate axes the metric terms vary along. There are also some parameters, mostly related to the internal grid generation, which are not alterable at run time because we simply have not gotten around to altering this. As grids rarely change between runs in an experiment, and often are read from a file instead of being internally generated, this does not seem to be a big problem. In the future, we will probably separate grid generation from model execution altogether, since the grid is often required for preprocessing input data. To see whether a specific parameter can be changed at runtime, grep for it in `HIM_parser.F90`; most parameters and options can be changed at runtime.

HIM makes very limited use of the F90 standard namelist. The only variables taken in as namelist variables are:

`output_directory` - the path to the directory for HIM-controlled output
`input_filename` - Either 'n' for a new run initialized as specified in `HIM_initialization.F90`, 'r' to (re)initialize from one or more restart files, or a list of filenames in which to find the variables required to restart a model. If this is a list of filenames, variables are initialized from the first file in which a variable matches the name that this variable uses in the restart files.
`restart_input_dir` - the directory in which to look for restart files.
`restart_output_dir` - the directory in which to write out restart files. This may be the same as `re-`

`start_input_dir`.
`parameter_filename` - the path to the file that is to be parsed at runtime to change parameter and option settings. The syntax of this file is the same as `HIM_init.h`.

Model output fields and frequency can be specified by editing the `diag_table`, an example of which is at `HIM_diag_table` in each of the examples' directories. For more information on the FMS diagnostics manager, click here [../src/shared/diag_manager/diag_manager.html].

4. Compile the source code and execute the runscript

HIM requires that NetCDF libraries be installed on users' platform. It is also desirable to have MPI libraries installed.

HIM code can be compiled and run with two memory allocation schemes: *STATIC* and *DYNAMIC*. All compile scripts provided here are for the *STATIC* option.

Experiments are based on the HIM Lima internal release. For increased efficiency, the `#define STATIC_MEMORY` compile option (set in `HIM_init.h`) allows HIM on the SGI Origins to run about twice the speed as dynamic memory allocation. The static option requires each processor to have identical sized domains, and for these domains to be specified at compile time. Hence, a bit of algebra is required prior to compiling the experiment. Specifically, `NXPROC` and `NYPROC` in `HIM_init.h` should be set so that their product is the desired number of processors, and these must evenly divide the number of grid points in each direction, `NXTOT` and `NYTOT` respectively.

For the *DYNAMIC* option one must edit HIM_examples/test_case/HIM_init.h and undefine `STATIC_MEMORY`. The FMS infrastructure will then attempt to decompose the domain.

In the case where MPI libraries are not installed on the users' platform, the HIM source code can be compiled without MPI by omitting the `cppDefs` value `-Duse_libMPI` in the example runscript. In this case the model can only be run on 1 processor and the user should undefine the compile options `PARALLEL_X` and `PARALLEL_Y` in HIM_examples/test_case/HIM_init.h. This is also a useful test if running on multiple processors is problematic. Be aware that the memory requirements of the more complicated models (global, him_sis) may preclude the running of the model on a single processor. The user should also remove *mpirun -np $npes* from the line that executes the model.

Under `ROOT/scripts` there is one script for both compile and run. All scripts have the name beginning with `run_him` followed by `test_case` name (`2gyre,DOME,` etc.). Before executing the script make sure to change the platform to the platform of your computer system. The platform is specified by:

```
      set platform = sgi # this is for sgi platform
```

and corresponds to the mkmf.template to be used. The mkmf.template can be found in the ROOT/bin directory.

Users may also want to change the following before starting compilation and execution:

```
      set npes = number of processors used in the run (with static memory allocation
                 this must be consistent with the values in HIM_init.h)
      set days = the length of the run in days (may be used to override the value set
                 in HIM_init.h or HIM_input).
      set months = the length of the run in months (may be used to override the value
                 set in HIM_init.h or HIM_input).
```

Users should also edit the run script to reflect the desired output paths.

5. Examine the output

To keep the runscript simple all output files of a model run will be in the work directory. There are three types of output files:

1. ascii file with `.fms.out` extension: the description of the setup of the run and verbose comments printed out during the run.
2. restart files in `RESTART` directory: the model fields necessary to initialize future runs of the model.
3. history files with `.nc` extension: output of the model, both averaged over specified time intervals and snapshots.

The ascii file contains everything written to the screen during model execution. The total time for model execution as well as the times of separate modules are reported here.

Users will see result files in NetCDF format. Postprocessing tools such as Ferret, ncview, grads or matlab can be used to view data in these files.

6. Brief description of HIM experiments

HIM is distributed with a set of test cases. These tests are taken from models used at GFDL for testing the numerical and computational integrity of the code.

Warning

These experiments are *NOT* sanctioned for their physical relevance. They are instead provided for the user to learn how to run HIM, and to verify the numerical and/or computational integrity of the code. *PLEASE* do not assume that the experiments will run for more than the short time selected in the sample run scripts.

`2gyre`: A two-layer, adiabatic, wind-driven double gyre in a basin with sloping sides. There are no thermodynamics in this case. This model is very small and can be easily run on a single processor workstation. It should provide the user with a basic experience of running HIM.

`DOME`: An entraining gravity current, driven by an inflow at the top of a slope. This is a coarse resolution version of one of the cases described in Legg et al., *Ocean Modelling*, 2005. This case has no surface forcing or surface mixed layer submodel, and only buoyancy as a thermodynamic variable, but it is otherwise a full ocean model.

`benchmark`: An idealized wind- and buoyancy-driven high-resolution model. This has all the physics of a full ocean model, but in a configuration that requires no input files. Versions of this have been used by NOAA for benchmarking computers. To change the resolution, only the number of gridpoints and timestep need to be changed. For convenience these have been gathered together in the `cppDefs` variable in the runscript.

`global`: A 1-degree, 48-layer ocean-only prototype for a global ocean-climate model on a tripolar grid. This is driven by the CORE forcing and uses restoring of surface properties to climatological values. *THIS HAS NOT BEEN FULLY TUNED, AND IS NOT DIRECTLY USEFUL FOR CLIMATE STUDIES!*.

`him_sis`: The same model as `global`, but coupled to GFDL's sea ice model (SIS). Running this model is somewhat different from HIM-only runs in that flow control is ceded to the standard GFDL coupler, and the HIM code is not used to specify the forcing. From a user's perspective, running this model is virtually identical to running a MOM4 model coupled to SIS.