HIM Frequently Asked Questions

William Cooke <`William.Cooke@noaa.gov`>

This page compiles a set of common questions that have arisen when starting to use HIM The questions are answered more fully in other parts of the documentation. They are included here in an abbreviated form to help people start using the model quickly.

Q:
Why did you rewrite HIM from C into Fortran? Isn't this an unusual direction for "progress"?

A:
The C code was very portable and efficient, but coupling it with GFDL's Fortran atmospheric models was awkward at best. The Fortran HIM code now has an identical coupling interface to MOM4, and is actively being developed as a part of a coupled model at GFDL. In addition, the Fortran HIM code is able to use the full FMS machinery for a variety of potentially machine dependent tasks, such as parallelization, I/O, and calender management.

In the conversion, we were careful to ensure that the equivalent Fortran- and C- codes gave bitwise identical answers. All of the HIM development at GFDL is now focused on the Fortran code.

The computational performance of the C- and Fortran- HIM codes are fairly similar when the Fortran code uses static memory allocation. The C- code is slightly faster on a single processor (due to better variable scope limitation), while the Fortran code scales to slightly higher processor counts due to the use of fewer barriers. The most noticeable performance difference is that the Fortran code often uses much more memory due to differences in the way that data is buffered and diagnostics are handled.

Q:
How do I specify HIM model parameters and options?

A:
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 runtime file is specified as the HIM_namelist variable `parameter_filename`. `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.

Q:
Why does HIM make such limited use of namelists?

A:
The Fortran namelist facility has very primitive error handling facilities and does not permit comments within the namelist file. By parsing its own file, HIM is able to check for syntax errors and the file is replete with comments that explain what each parameter or option means, along with its expected units.

Q:
How do I specify initial conditions?

A:
Initial conditions are typically set by editing `HIM_initialization.F90` to specify which files and variables to read or analytic expressions. Alternately, HIM can be (re)initialized from a restart file, or from

a set of NetCDF files with the same core variables as are in a restart file (u, v, thicknesses, etc.). Some of the variables in the restart files can be approximately reproduced by the model and are not required in a new run. (For example, the average velocity over a time-step is approximately its value at the end of the time step.)

The namelist variable `input_filename` determines where the initial conditions come from. It is '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.

Q:

How do I specify surface forcing?

A:

Surface forcing for ocean-only models is specified by editing `HIM_surface_forcing.F90`. Here, the subroutines `wind_forcing()` and `buoyancy_forcing()` are used to set the two types of surface forcing fields. The ocean components of coupled models take all of their forcing fields from the ice- or atmospheric-component.

All of the forcing fields are passed around the model in a structure of defined type `forcing`. This structure contains pointers to all of the fields that are in use, and unassociated pointers to unused options. This construct allows HIM to use a variety of reasonable combinations of forcing fields. In this structure, all fluxes of substances are positive downward. For example, HIM can accept just buoyancy forcing, or it can accept a combination of heat and fresh water fluxes. The fresh water fluxes can be expressed as `(Precipitation - Evaporation)`, or `Precipitation` and `(-1*)Evaporation`, or `Liquid Precipitation`, `Frozen Precipitation`, and `(-1*)Evaporation`. See the definition of the `forcing` type in `HIM_variables.F90` for all the available options.

Q:

How do I specify which fields are output and how often?

A:

HIM uses the FMS diagnostics manager to control its output. A file named `diag_table` is parsed to determine which fields are output. This file has two sections: (1) a list of the files to be written along with their output frequencies, and (2) a list of the variables that go into the files. Comments at the end of `diag_table` describe the syntax. A single variable can be put into multiple files by listing it on multiple lines. All of the HIM examples include sample copies of `diag_table`. Most of the available diagnostics are in these sample files, either as active entries or entries that are commented out.

Q:

Are the compile and run scripts platform dependent? What do I need to change when I go from one platform to another?

A:

In GFDL, we try our best to make the scripts independent from platforms as much as possible. In theory, all platform specifics are contained in Make templates (`mkmf.template.platform`). When users set platform (sgi, ibm, ifc ...) the scripts will pick the right template.

In reality, users encounter various problems related to implementation on different platforms. Users may need to modify the source code and/or scripts. If not, please report your problems to us and we try our best to address your problems.

Our limitation is that in GFDL currently we have only SGI, Altix and Beowulf (using IFC). HIM code has been tested extensively on these platforms. For experience running HIM on other platforms we rely on contributions from external users. For the benefit of the HIM community please report problems and or solutions related to your platform.

Q:

How do I set the length of run?

A:

For historical reasons there are two ways to control the length of the run. In the HIM_input file you will see:

```
     #define DAYMAX 20
                                 !    The final day of the simulation.
```

You can change DAYMAX to the number of days you want the simulation to run. In addition with this option, you can set the wall-clock time that you want HIM to run before restarting itself by setting the value of MAXCPU in the HIM_input file.

In the runscript you may see:

```
     &ocean_solo_nml
       set days = 20
       set months = 0
```

You can change days and months to any number you want. If ocean_solo_nml (or coupler_nml in the case of a coupled run) is present this will override the use of DAYMAX in the HIM_input file. In the case of a coupled model run, the namelist coupler_nml is required.

Q:

How do I set the number of processors?

A:

In the runscript you see:

```
     set npes = 15
```

In this example, there are 15 processors, you can set npes to any number your system can afford. However, if using the STATIC option, the number of processors must be the product of NXPROC and NYPROC in HIM_init.h

Q:

What is STATIC_MEMORY option, how do I use this option?

A:

Based on our experience with SGI Origins at GFDL, the compiler is performing an excessive number of array copies if all the arrays are dynamically allocated. The solution is to allocate arrays at compile time. To select the static memory option, include the line "#define STATIC_MEMORY" in the copy of `HIM_init.h` that you compile with.

Requirements of static option: each processor should have the same number of grid points as all other processors. For example, if your global grid is 100x100 you can not have 13 processors but you can have 2, 4, 8, 10 ... processors. This is not a fundamental requirement, but is a limitation of the current FMS infrastructure.

Q:

How can I change the time step of the model?

A:

In the file HIM_input look for DT. You will see:

```
     #define DT 1800.0
                                 !    The time step, in s.
```

This is time step of ocean model measured in seconds. Note that a longer time step may violate CFL conditions and the model may bomb. You may also need to change the barotropic time step.

```
      #define DTBT 450.0
                               !    The barotropic time step, in s.
                               !  DTBT is only used with the split
                               !  explicit time stepping.
```

In the case of a coupled model run, look for the coupler_nml namelist in the runscript. "dt_ocean" corresponds to the frequency with which the ocean is called.

```
       dt_ocean = 1800
```

Q:
How do I view the model output?

A:
Output files are left in the $workdir directory in NetCDF format. Some common tools working with NetCDF are ncview and ferret.

Q:
I can not compile `memuse.c`, is anything missing here?

A:
It is possible that in `mkmf.template` the following is missing: CFLAGS = -D__IFC

If it is the case, you need to add that to `mkmf.template`.

Q:
What should I do to run a test case using restart data?

A:
Basically, you need to use restart data instead of initial condition data. If you are using the runscripts provided by the **HIM** release. Do the following:

```
      Replace the data file pointed to by
        set input_data   = $cwd:h/data/initial_condition_file

      by the following :

        set input_data   = restart_file

      where
        initial_condition_file is the file provided via the data server.
        This is only necessary for the global and coupled him_sis runs presently.

      and
        restart_file is a tarred combination of the restart file(s) written to
        restart_output_dir in HIM_input_nml (typically "RESTART/").
```

Q:
How can I debug HIM?

A:
The debugging tool used at GFDL is Totalview. To use this tool the user must:

```
     - compile the code with -g option in FFLAGS
     - insert "make localize" in the runscript so that all source code will be copie
```

If you do not have Totalview, then the old-fashioned method of adding print statements is the suggested method for debugging.

Q:

What is MPI and how can I use it?

A:

At GFDL we use the Message Passing Interface to allow us to run on multiple processors. If you do not have MPI, you will need to remove -Duse_libMPI from the line that creates the Makefile. But you will only be able to then run on a single processor. The simpler test cases have been tested on a single processor, but the global model and HIM_SIS coupled model probably have too large a memory footprint to run on a single processor.

Q:

How do I know if two runs using different number of processors produce identical results?

A:

It is good idea for each test case to check if:

```
    - model results reproduce across different sets of processors
    - model results reproduce across restart data
```

When using different sets of processors, the checksums printed out by the HIM model will typically not be identical.

```
On 1 processor
 HIM Day    80.000   5760: En 3.828572E+12, Vol 5.109350983582E+15, Salt 0.000000000
Day    80.000 Total Energy:  3.82857E+12
 Total Energy: 428BDB4561FB5E69  3.8285719796278013E+12

On 8 processors
HIM Day    80.000   5760: En 3.828572E+12, Vol 5.109350983582E+15, Salt 0.0000000000
Day    80.000 Total Energy:  3.82857E+12
 Total Energy: 428BDB4561FB5EF6  3.8285719796278701E+12
```

This is due to the order of summation across processors by the model. The differences should only occur in the final few digits of the hexadecimal and verbose energy line (the line beginning "Total Energy:" above). However the output files should be identical and can be compared using the `cmp` command.

In the 'reproduce across restart' cases users can rely on the checksums that are printed at the end of each run (output file `fms.out`). A NetCDF tool `ncdiff` can also be used to view differences of two `.nc` files.

Q:

Should I care about the contents of the fms.out file

A:

HIM has plenty of material written to `stdout` (which is what is written to `fms.out`). The user is strongly recommended to become familiar with the contents of `fms.out`, especially when building a new model experiment.

Q:

How seriously should I take the HIM-test cases?

A:

The HIM-test cases are not well tuned models. Hence, they generally will not result in physically meaningful simulations. Additionally, the models may crash if run for longer than suggested in the distributed run scripts. That is, the test cases are supported *ONLY* for testing the integrity of the code on various platforms (i.e., does the code compile and run for a few time steps?). *PLEASE PLEASE* do not take the test cases and write papers based on them. We are not providing models -- instead we are providing code.

Q:

What is the significance of the time at the top of the `diag_table`? How does this time interact with the time for the model? What happens if there are inconsistencies between the `diag_table` time and the "date_init" entered into `ocean_solo_nml` or "current_date" entered into `coupler_nml`?

A:

Let's call the time at the top of `diag_table` T1 and the time entered in namelist T2. The time at the top of the `diag_table` (T1) is used as starting time of time axis in all output files that have time axis. When you open any output file and look for variable Time, you will see:

```
     Time:units = "hours since T1"
```

(assuming unit is hours).

T2 is the initial time for running the model, all fields cannot be written to output files sooner than T2. Since time can not be negative you should have T2 >= T1. In practice, to keep values in time axis from being exceedingly large, you should have T2=T1.

Q:

How do I change the resolution of the test cases?

A:

The various test cases can be modified in the following ways. The parameters mentioned below all reside in the appropriate HIM_examples/test_case/HIM_init.h file. Of course, we cannot guarantee that any configuration that you may apply will actually work.

2gyre : The domain here can be changed by varying the parameters LENLAT, LENLON, NXTOT, NYTOT.

DOME : In order to reproduce the 10km resolution of the Legg et al paper model runs, one should change the number of model grid-points. This is accomplished by setting NXTOT to 150 and NYTOT to 70. This will then create a 10km resolution from the domain lengths LENLON and LENLAT. One should also change the time steps (DT, DTBT) in approximately an inverse ratio.

Benchmark : One can easily change the size and resolution of the benchmark cases by changing the variables NXTOT, NYTOT and DT, DTBT in the cppDefs section of the runscript.

global : The resolution is tied to information in the grid_spec.nc file supplied with the model. Therefore it is not a simple task to change the resolution of this model.

him_sis : The resolution is tied to information in the grid_spec.nc file supplied with the model. Therefore it is not a simple task to change the resolution of this model.