Users Manual for CSU Geodesic Shallow-Water Model
doe_ca/swm/doc/users_manual.pdf
April 12, 2004

1. Obtaining the SWM

The SWM is one component of a complete coupled system model. To obtain the full shallow-water model (*swm_csu.tar*), contact Kelley Wittmeyer (*kelley@atmos.colostate.edu*) and ask her for the username, password, and website to download the *swm_csu.tar* file.

Point your browser to that website, login, and download the *swm_csu.tar* file. On your local machine, make a directory called *doe_ca*, extract the tar file (*tar -xvf swm_csu.tar*) inside the *doe_ca* directory. Under *doe_ca* there will be four directories: *utilities*, *esmf, lib*, and *swm*. The path in the build system assumes the top-level directory is *doe_ca* so it is important to extract the *swm_csu.tar* directly under *doe_ca.*

2. Compiling the SWM

a. The build

Go to *utilities/compile* and type “*make DEBUG=true*”. This should produce a file *utilities.lib* in the *doe_ca/lib* directory. Following the utilities build, go *swm/compile* and type “*make DEBUG=true*”. This build will compile the SWM source code, link to the *lib/utilities.a* file, and link to system level library files. The result of the build is the creation of an executable called “*swm*” that resides in the local *swm/compile* directory. In this case, the swm executable is compiled in debug mode. If the user wishes to recompile the model with a higher optimization, both the utilities and swm compile directories should be cleaned (with “*make clobber*”) and recompiled with “*make*”.

b. Compilation directives

Compilation directives are included in *utilities/Makefile.library* for the utilities build and *swm/compile/Makefile.model* for the swm build.

For a full description of the utilities software see the following document: (Don Dazlich). The utilities build includes the following directives that the user may wish to alter:

```
DMPI          = -Dmpi=0
DPARALLEL_IO    = -Dparallel_io=0
```

DMPI controls whether message passing is enabled in the utilities library. The default for *DMPI* is *0* (false) for a single-processor library. *DPARALLEL_IO* controls whether I/O is done in serial (single processor) mode or in parallel model. We suggest that users leave the default setting of *0* (false) in place. Note that once the utilities library is built, it does not have to be re-compiled unless the user wish to change the compile directives, change the level of optimization, or change the source code.

The swm build includes the following directives that the user may wish to alter:

```
RESOLUTION     = 0002562
DNUMSBDMNS     = -Dnumsbdmns00010
DNPE           = -DNPE_COMP=1
DMPI           = -Dmpi=0
DPARALLEL_IO   = -Dparallel_io=0
DVISCOSITY     = -Dviscosity=0
```

DVISCOSITY controls whether a del4 diffusion is applied to the momentum equation with a coefficient set in the call to *dissipation* from *swm/source/driver.F*. The default is *0* (false). The *DMPI* and *DPARALLEL_IO* have the same meanings as in the utilities build and **MUST** be set to the same values as used in the utilities build. In addition, the user specifies the model resolution *(RESOLUTION*) as *0002562*, *0010242*, *0040962*, or *0163842*. Other resolutions, both higher and lower, are available, but the repository only includes startup files for these resolutions. If *DMPI* is set to 0, then *DNUMSBDMNS* must be set to *00010* and *DNPE* set to *1*; this states that the single cpu executable will run on one processor and the global grid will be composed into 10 subdomains. If *DMPI* is set to one (in both the utilities build and the swm build), then *DNPE* states the number of processors that will be used and *DNUMSBDMNS* states the total number of subdomains used to decompose the global grid. Below is a sample of valid *DNPE* and *DNUMSBDMNS* combinations:

DNPE_COMP=2	*Dnumsbdmns00010*
DNPE_COMP=4	*Dnumsbdmns00040*
DNPE_COMP=4	*Dnumsbdmns00160*
DNPE_COMP=8	*Dnumsbdmns00040*
DNPE_COMP=16	*Dnumsbdmns00160*
DNPE_COMP=32	*Dnumsbdmns00160*

Many other valid combinations exist. The requirement is that the number of subdomains be $(10) \bullet (4)^n$ where n starts at 0 and the number of processors must divide evenly into the number of subdomains. In general, users should choose the lowest number of subdomains possible for a given number of processors.

c. Trouble shooting

1. The make system for the SWM is set up to automatically detect the following architectures: IBM SP, SGI, and Macintosh (Absoft). If you are building the SWM on one of these architectures, you will hopefully not have to alter the make system. If you are building the SWM on another

system, you will have to edit the *Makefile.host* files in *utilities/compile* and *swm/compile* and add your system to the list of supported architectures.

2. On SGI machines the build system works best using gmake. Using "*gmake*" instead of "*make*" can fix most build problems on SGI machines.

3. The SWM requires several system level libraries. These include netCDF and MPI. If your system has these libraries, but you are getting errors at the end of compilation due to "files not found," you will have to alter the *Makefile.host* files and correct the path to these libraries. These will be modified in the *LDLIBS* line for the architecture you are using.

3. Running the SWM

At this point an executable has been created and the user has moved that executable into the correct run directory. For example, a swm built at a resolution of 2562 should be moved into the run.0002562 directory. The following run-time variables can be set by the user in *run.*/data/namelist*:

```
0001.00  ! tau_end_day: the number of days to integrate forward
002.00   ! tau_end_hour: the number of hours to integrate forward
0100.00  ! dt (seconds): the time step
F        ! l_restart: start from a restart (use only false at this time)
0864     ! n_restart: write restarts every n_restart steps
0036     ! n_output: write output every n_output steps
0036     ! n_diagnostics: write diagnostics to std_out every n_diagnostics steps
```

Once these parameters are specified, the user launches the executable. If the system was compiled with *-Dmpi=0*, use "*swm > log*". If the system was compiled with *-Dmpi=1* use "*mpirun -np ? swm > log* where "*?*" matches the value of *NPE* used in *swm/compile/Makefile.model*.

The model is set up for shallow-water model test case #5 which includes a 2 km conical mountain inserted into a geostrophically-balanced zonal flow at t=0. The simulation is characterized by the excitation of gravity waves and Rossby waves with a reasonable degree of nonlinearity occurring in the vicinity of the mountain. The initial conditions are specified in *swm/source/initialize.F.* High resolution spectral solutions that serve as a control reference solution are available on request.

4. Visualizing SWM results

Two types of output are written. One form of output is written to *run.*/dx/output*. This output is visualized with OpenDX and only users familiar with OpenDX should work with this data. The other data is written to *qp_output* in netCDF format. This data is readily viewable with IDL combined with the (free) CSU visualization package EZPLOT. If you have IDL, contact Kelley Wittmeyer for a copy of EZPLOT. To use EZPLOT do the following: start IDL, type EZPLOT, on the

EZPLOT GUI click “CSU GCM Data” and choose “GeoDesic File” and open any file in *run.*/ qp_output*. EZPLOT will read in the data for all points at all time levels and provide a point-and-click GUI to visualize the data.

The netCDF files are self-describing; the entire grid information is contained in the files. A “*ncdump*” of a typical files yields the header information listed in Appendix A. The vector listing of *grid_center_{lat,lon}* and *grid_corner_{lat,lon}* provides a complete description of the grid. There is a one-to-one correspondence between a grid center and the variable data. These netCDF files should be viewable using many of the currently available graphics packages.

Appendix A: netCDF header information for the SWM

```
netcdf hswm_d000000p000.g2 {
dimensions:
	time = UNLIMITED; // (26 currently)
	char_len = 10;
	grid_cells = 2562 ;
	grid_size = 2562 ;
	grid_corners = 6 ;
	grid_rank = 1 ;

variables:
	double time(time) ;
		time:quantity = "time" ;
		time:units = "days since 1-1-1" ;
		time:calendar = "noleap" ;
	char char_time(time, char_len) ;
		char_time:format = "mm/dd/yyyy" ;
	double start_period(time) ;
		start_period:long_name = "start of averaged period" ;
		start_period:units = "days since 1-1-1" ;
	double end_period(time) ;
		end_period:long_name = "end of averaged period" ;
		end_period:units = "days since 1-1-1" ;
	double period_length(time) ;
		period_length:long_name = "length of averaged period" ;
		period_length:units = "days" ;
	double grid_center_lat(grid_cells) ;
		grid_center_lat:units = "radians" ;
	double grid_center_lon(grid_cells) ;
		grid_center_lon:units = "radians" ;
```

```
long grid_dims(grid_rank) ;
double grid_corner_lat(grid_cells, grid_corners) ;
        grid_corner_lat:units = "radians" ;
double grid_corner_lon(grid_cells, grid_corners) ;
        grid_corner_lon:units = "radians" ;
float thickness(time, grid_cells) ;
        thickness:long_name = "thickness of fliud layer" ;
        thickness:title = "thickness of fliud layer" ;
        thickness:units = "meters" ;
        thickness:positions = "center" ;
        thickness:missing_value = 9.9999996e+35f ;
float height(time, grid_cells) ;
        height:long_name = "surface height of fliud layer" ;
        height:title = "surface height of fliud layer" ;
        height:units = "meters" ;
        height:positions = "center" ;
        height:missing_value = 9.9999996e+35f ;
float relative(time, grid_cells) ;
        relative:long_name = "relative vorticity of fluid" ;
        relative:title = "relative vorticity of fluid" ;
        relative:units = "1/s" ;
        relative:positions = "center" ;
        relative:missing_value = 9.9999996e+35f ;
float divergence(time, grid_cells) ;
        divergence:long_name = "divergence of fluid" ;
        divergence:title = "divergence of fluid" ;
        divergence:units = "1/s" ;
        divergence:positions = "center" ;
        divergence:missing_value = 9.9999996e+35f ;
float absolute(time, grid_cells) ;
        absolute:long_name = "absolute vorticity of fliud layer" ;
        absolute:title = "absolute vorticity of fliud layer" ;
        absolute:units = "1/s" ;
        absolute:positions = "center" ;
        absolute:missing_value = 9.9999996e+35f ;
float kinetic_energy(time, grid_cells) ;
        kinetic_energy:long_name = "kinetic energy of fluid" ;
        kinetic_energy:title = "kinetic energy of fluid" ;
        kinetic_energy:units = "1/s" ;
        kinetic_energy:positions = "center" ;
        kinetic_energy:missing_value = 9.9999996e+35f ;
float tracer_1(time, grid_cells) ;
        tracer_1:long_name = "tracer field #1" ;
        tracer_1:title = "tracer field #1" ;
        tracer_1:units = "tracer per unit thickness" ;
        tracer_1:positions = "center" ;
```

```
                tracer_1:missing_value = 9.9999996e+35f ;
        float tracer_2(time, grid_cells) ;
                tracer_2:long_name = "tracer field #2" ;
                tracer_2:title = "tracer field #2" ;
                tracer_2:units = "1/s" ;
                tracer_2:positions = "center" ;
                tracer_2:missing_value = 9.9999996e+35f ;
        float tracer_3(time, grid_cells) ;
                tracer_3:long_name = "tracer field #3" ;
                tracer_3:title = "tracer field #3" ;
                tracer_3:units = "tracer per unit thickness" ;
                tracer_3:positions = "center" ;
                tracer_3:missing_value = 9.9999996e+35f ;
        float tracer_4(time, grid_cells) ;
                tracer_4:long_name = "tracer field #4" ;
                tracer_4:title = "tracer field #4" ;
                tracer_4:units = "tracer per unit thickness" ;
                tracer_4:positions = "center" ;
                tracer_4:missing_value = 9.9999996e+35f ;
        float tracer_5(time, grid_cells) ;
                tracer_5:long_name = "tracer field #5" ;
                tracer_5:title = "tracer field #5" ;
                tracer_5:units = "tracer per unit thickness" ;
                tracer_5:positions = "center" ;
                tracer_5:missing_value = 9.9999996e+35f ;
        long interp_indx(grid_size) ;

// global attributes:
                :calendar = "noleap" ;
                :institution = "Colorado State University" ;
                :history = "Created: 04/02/2004 at 14:03 -0700 GMT " ;
                :run = "SWTC#5" ;
                :grid = "geodesic" ;
                :version = "0.6" ;
                :total_grid_size = 2562 ;
```