Getting the code to work: Proof of concept in an interactive shell.

1) Codes are executed with "run scripts." These are shell script text files that set up the individual runs and execute the code. The scripts will seem a bit overly complicated to use at first; this is so they can be submitted as "jobs" to the queue. (This is how you run the model for any significant period of time, and will be discussed later.)

There's a two step process to running the model. I've generated a shell script, "`create_runscript.csh`" that creates the run script to run the cores. You can create run scripts for all three cores with this one script by changing the input parameters.

First we need to understand how the model works. Any run of the GCM requires input files to tell the code the parameter values for the integration (i.e. what is the rotation rate of the planet, the resolution, etc.), the output you'd like to have (i.e. zonal wind, surface pressure, etc.), any tracers you'd like included in the run (a tracer is something advected by the flow, like water vapor or pollution), input topography files, and so forth. For a comprehensive atmospheric GCM like AM2.1 or AM3, the input files take up gigabytes of space! The parameters governing the simulations are stored in the directory:

```
student_models/run_parameters/simulationID
```

where simulationID identifies the run you'd like complete. I found it works well to have one place where you put all the input parameters. This way you can always go back and see exactly what you specified for the simulation. (It turns out that you don't have to specify most parameters. If you don't specify them, the model picks the default value -- you may have to look at the fortran source to find out what the default is.)

We have three different dynamical cores, and I created three default runs, one for each. (I've tested the spectral models — if the others don't work,

contact me, as we made need to upgrade the compile scripts.) These allow you to run the standard Held Suarez (1994) simulations (hence the integration IDs starts with "nrelax" for Newtonian-relaxiation). The models are described on the models webpage, where you found this document. Here are the three models. I give the time step that you need, for reference.

`nrelax_t42l20e_control`

This pseudo-spectral model is based on a spherical harmonic decomposition, but advection is nonlinear, hence the "pseudo-" part!) The time step is 1800 seconds, and the number of processors should divide 64 (for this resolution). The t42 refers to resolution, triangular truncation to wavenumber 42. The corresponding grid is 128 longitudes by 64 latitudes. `l20e` means there are 20 vertical levels which are "e"venly spaced. "hs" for Held-Suarez!

`nrelax_bgrid_n45l20e_hs`

This model uses finite difference numerics on a bgrid (an offset grid for winds, pressure fields, etc. There are different ways to do this; as the name implies, this is version b). The time step is 1800 seconds. I'm actually not sure about the number of processors (don't use this core much, as it's slow), so you'll need to experiment! n45 means there are 45 latitudinal grid points between the equator and pole.

`nrelax_n45l24_hs`

This model uses finite volume numerics -- this is the core used in GFDL's AM2.1. The time step is 1800 seconds, and the number of processors must divide 90. n45 again refers to 45 levels between equator and pole. The grid here is anisotropic; the resolution is finer in latitude than in longitude. This gives you more bang for your computational buck, as baroclinic eddies tend to be more sensitive to meridional resolution than zonal resolution, particular when they are only marginally being resolved.

Note that we also have the cubed sphere finite volume core used by GFDL's AM3 (and soon to be the core for the National Weather Service's prediction system). This isn't as user friendly, but you can contact me if you'd like to use it. And for good measure, we also have a number of NCAR dynamical cores running here!

2) To set up a run, you must specify these things in "create_runscript.csh," a c-shell script found in your jobs directory:

`/home/nyuID/student_models/jobs/create_runscript.csh`

Edit this with your favorite text editor (say, vi or emacs), for example:
> emacs `/home/nyuID/models/jobs/create_runscript.csh &`
The & ensures that your terminal stays active. Otherwise, you can't use it again until you are done with emacs. If you copied over my .bashrc shortcuts, you can fire up emacs this way:
> em `/home/nyuID/student_models/jobs/create_runscript.csh`
The shortcut takes care of the & for you!

Here's what the file looks like, and the things you need to specify to run the model. Make sure to specify your nyuID towards the end of the list.

```
#!/bin/csh -f

# Set the run parameters here.  This script will create a job script.
#-----------------------------------------------------------------#
# Define the experiment and NYU/NCAR variables, if applicable

set job_id     = nrelax_t42l20e_control
# simulation name; there should be a corresponding run_parameters directory

set t_start    = 0  # day to begin (0 for cold start)
set t_end      = 10  # final day of integration
set dt_atmos   = 1800    # time step of model (seconds)
set delta      = 5    # time for each run (days)

if ($delta == 0) @ delta = $t_end - $t_start
```

```
set npes_per_node   = 16  # number of processors per node
set n_nodes         = 1      # number of nodes requested
set wall_clock_max  = 1:00:00   # wall clock max (in hours:minutes:seconds)
set memory          = 32    # memory required (in GB)


set model_numerics  = spectral  # spectral (for the pseudospectral model)
                        # fv (for S.J. Lin's finite volume core)
                        # bgrid for bgrid core
                        # csfv for cubed sphere finite volume core

set model_radiation = nrelax
      # nrelax for Newtonian relaxation
      # gray   for Gray radiation (Frierson 2006)
      # rrtm   for RRTM radiation

set source_branch     = ""  # this allows you to switch to a new branch
     # of the code, which will have a different executable.  Default is ""
     # In general, leave this blank unless you're using the spectral model
     # before we added the gravity wave drag option, in which case set
     # it to "_pre_gwd"
set experiment_type  = iterate
     # 'iterate' to run the model along, sequentially
     # 'life_cycle' to run a life cycle with specified initial
     #              conditions (requires namelist parameters,
     #              and initial_conditions.nc files.)
     # 'ensemble' to a series of ensemble runs

# the cubed sphere core needs a different script, and currently only
# can be iterated
if ( "$model_numerics" == "csfv" ) then

  if ( "$experiment_type" == "iterate" ) then
     set experiment_type = iterate-csfv
  else
     echo $model_numerics model can only be iterated at this time
     exit
  endif

endif

set platform         = nyu  # 'nyu' for NYU Dell cluster
                            # 'tempest' for my machine
                            # 'ibm' for NCAR machines
set user             = epg2
```

Change epg2 to your NYUID! You should not have to change anything below this in the script.

In time I will try to explain what all these parameters allow you to do.

3) Run the model interactively.

I'll first walk through the run with settings specified above. These will run the finite volume model for 10 days, from a cold start, restarting the model every 5 days. A small run like this is appropriate for running the core interactively. To do this, request an interactive session on a compute node:

```
> is8
```

This command is an alias from the sample.bashrc. You can also type the full command:

```
> srun --nodes=1 --cpus-per-task=8 --mem=40GB --time=4:00:00 --x11 --pty /bin/bash
```

Now you can see why aliases can save you time! We are asking for 1 node with 8 CPUs available, and 40 GB of memory, for up to 4 hours.

We are now on a different part of the HPC system, a compute node. You'll find that you can no longer see /archive/[first initial]/nyuID any more. However, now you can run parallel codes.

Go to your jobs directory.

```
> cd /home/nyuID/student_models/jobs
```

I assume you edited the create_runscript.csh file so it's ready to go. If not, do so now. For this interactive session, make sure there are less than 8

processors requested, as you only have access to 8 on the compute node. Then execute create_runscript.csh to produce the actual run script. The ./ part of this command tells your shell that the script is right here, in your current working directory.

```
> ./create_runscript.csh
```

The script should say that it's produced a job file, and tell you it's name (in this case, job.nrelax_t42l20e_control_d00010). It will always give the name of the simulationID, and the end date of your run.)

Unfortunately we need to modify one line to make it work interactively. (The script is set to work in the queue, where you should always run the model.) To fix it, you need to edit one line, using emacs, vim, or your favorite text editor. Change:

```
  srun --ntasks=$SLURM_NTASKS_PER_NODE ./fms.x
```

to be just:

```
   srun ./fms.x
```

Now execute the script. (Replace what I have here with the name of your script, if you picked something else.)

```
> ./job.nrelax_t42l20e_control_d00010
```

Somethings should happen. Text will fly by. With hope you won't see error messages, or indications that the model crashed.

When it's done, look for output. The script above should produce two new directories.

```
> ls /scratch/nyuId/model_output/
```

```
nrelax_t42l20e_control_d00005
nrelax_t42l20e_control_d00010
```

Here I've shown the two directories that appeared for us. (If you have other existing runs, they will be there too!)

Look inside these directories to find the output. We'll look at the second one, but both will appear the same.

```
> ls /scratch/nyuId/model_output/nrelax_t42l20e_control_d00005
atmos_average.nc  diag_table   fms.x  input.nml         RESTART
atmos_daily.nc    field_table  INPUT  logfile.0000.out  time_stamp.out
```

The output files from the model are atmos_average.nc and atmos_daily.nc. The first contains time mean values over the full integration. The latter contains daily output, instantaneous values written out once per day. Use ncview to look at one. (I set up the alias "nv" in sample.bashrc, if you don't have it, use the second line.)

```
> nv /scratch/nyuID/model_output/nrelax_t42l20e_control_d000010/atmos_daily.nc
```

or

```
> ncview /scratch/nyuID/model_output/nrelax_t42l20e_control_d00005/atmos_daily.nc &
```

ncview pops up a blue window. Click on variable under the middle box "Var:" To start, try ucomp, the zonal wind, u. Another window should pop up with a colorful plot with continental outlines. If you've gotten this far, you've run a GCM!!!!! Your output, however, may look pretty boring; fields that are practically uniform. This is because you've only run the core for 4 days. The model starts from an isothermal (uniform temperature) state, and takes about 200-300 days to "spin up" to a realistic state! The continental outlines are just for a sense of scale -- there are no continents in

this model, and ncview just puts them there by default. There is on ncview the PDF on model analysis.

Other important files are found in the subdirectory RESTART.

```
> ls /scratch/nyuId/model_output nrelax_t42l20e_control_d00005/RESTART

atmos_model.res  atmosphere.res.nc  spectral_dynamics.res.nc
```

These files may look a bit different if you run a different core (i.e. bgrid / spectral) but the presence of files here always indicates that the model ran successfully. Sometimes, however, you won't find files here. That means it crashed. There may still be some output files, like atmos_daily.nc, as the core will write out all the date up until when it crashed.

4) More general advice about the runs.

As described above, you must always specify a simulationID in create_runscript.csh, which identifies which parameters you would like to use. You also specify the time step for the integration, and how long the integration should run. The model output will always appear in directories like this:

```
/scratch/nyuID/model_output/simulationID_d[end_date_of_integration]
```

Even though everything looks the same in each directory, the model output should be marching forward in time. You can verify this by using ncview. Note the first date of the integration shown at the top. The cores begin integrating at 00:00:00 1-Jan-2001. The daily output is written every 24 hours, so in your first run it'll start at 00:00:00 2-Jan-2001. Confirm that in later output directories that this date continues to march forward, and doesn't reset back again!

5) How to restart.

Say you have run the model for 10 days, as described above, but would like to run it more. All you need to do is edit create_runscript.csh. Keep everything the same except the start and end dates. To restart, set the start date to the final date of the previous run, i.e.

```
set t_start    = 10             # day to begin (0 for cold
start)
set t_end      = 20       # final day of integration
```

Assuming you haven't changed delta from above, this will cause the model to begin integrating at day 10, run 5 days, restart, and then run 5 days more, for a total of 20 days of integration. The restart option assumes that you have left the previous run in your scratch output directory. All it needs are the restart files. In this specific case, it will look for:

```
/scratch/nyuId/model_output/nrelax_t42l20e_control_d00010/RESTART
```

and the files inside it that we listed above. It's okay to move the output files (atmos_daily.nc and atmos_average.nc) where ever you'd like, but if you want to restart the model, make sure that this directory is put back in the appropriate directory of: /scratch/nyuId/model_output/

6) Trouble shooting.

Many things can go wrong, probably many more that I could ever anticipate. Here are some common reasons for the script failing in the past.

Errors when you try to execute `./create_runscript.csh`
a) Shell scripts can be very finicky. When you set an option, make sure there is space between the equals signs
correct: `set t_start = 4`
incorrect: `set t_start =4`

Model does not run when you execute the job script (that is, when you type `./job_simulationID_d[end_date]`

a) Make sure that the numerics option matches the type of model you'd like to run. If you specify `n45l24_hs`, this is a finite volume core. If you set the option `model_numerics = spectral`, the model will fail.

b) The script checks to make sure that you haven't already run the model. If you specify a run that will produce a directory that already exists, it will stop before executing. It should produce an error message to this effect, though, telling you what was wrong.

c) Is this a restart? Are the restart files in the right place? If not, the script should produce an error message that it can't find the restart files.

d) Did you pick the right number of processors? The model will fail in the start up phase if it can't appropriately partition atmosphere between the number of processors that you have specified.

Now, you'll want to learn how to run the model with the jobs queue, where you can run it for much longer. You're ready to move to the next guide, submitting_model_runs_to_the_queue.pdf.