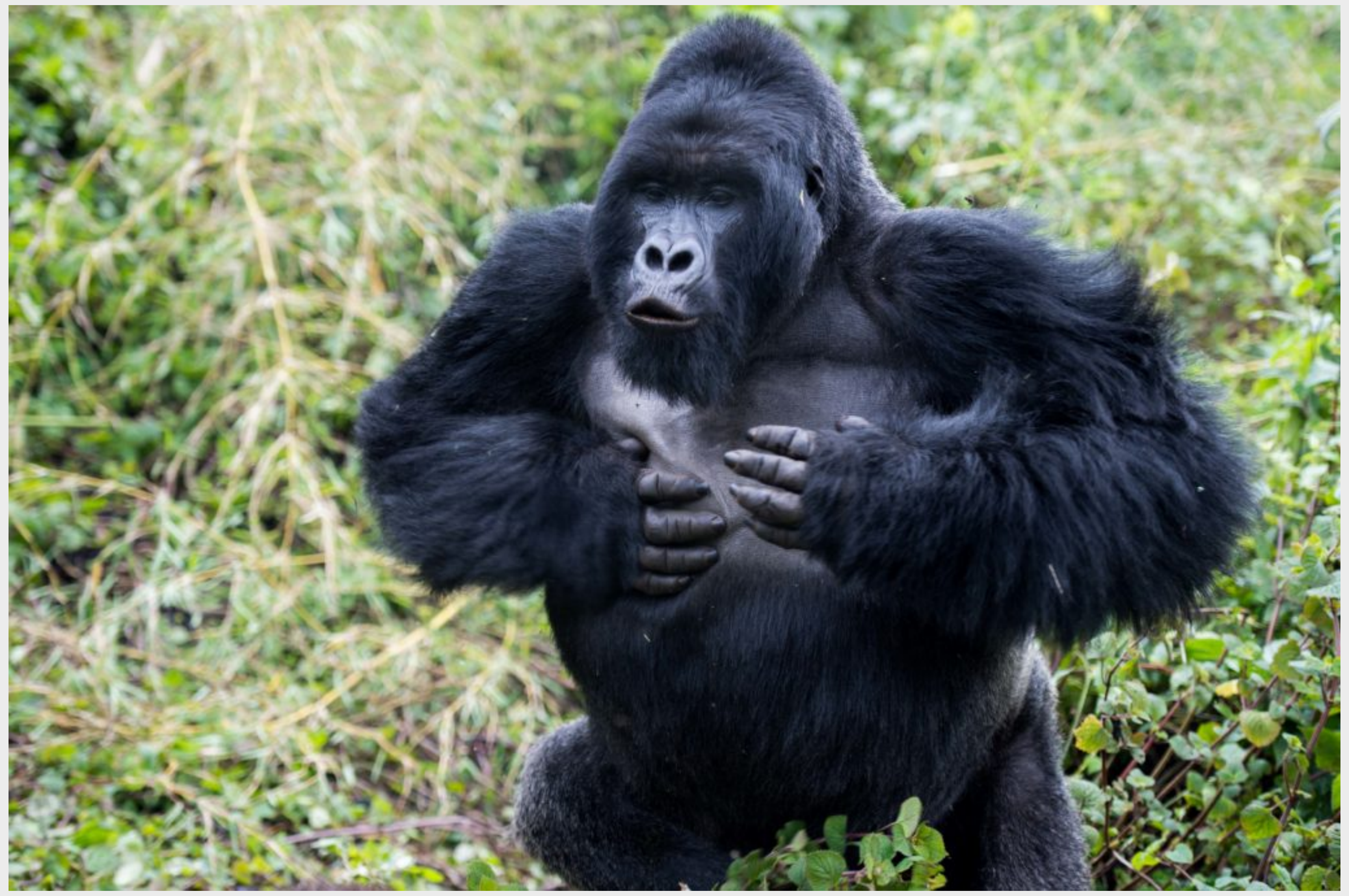


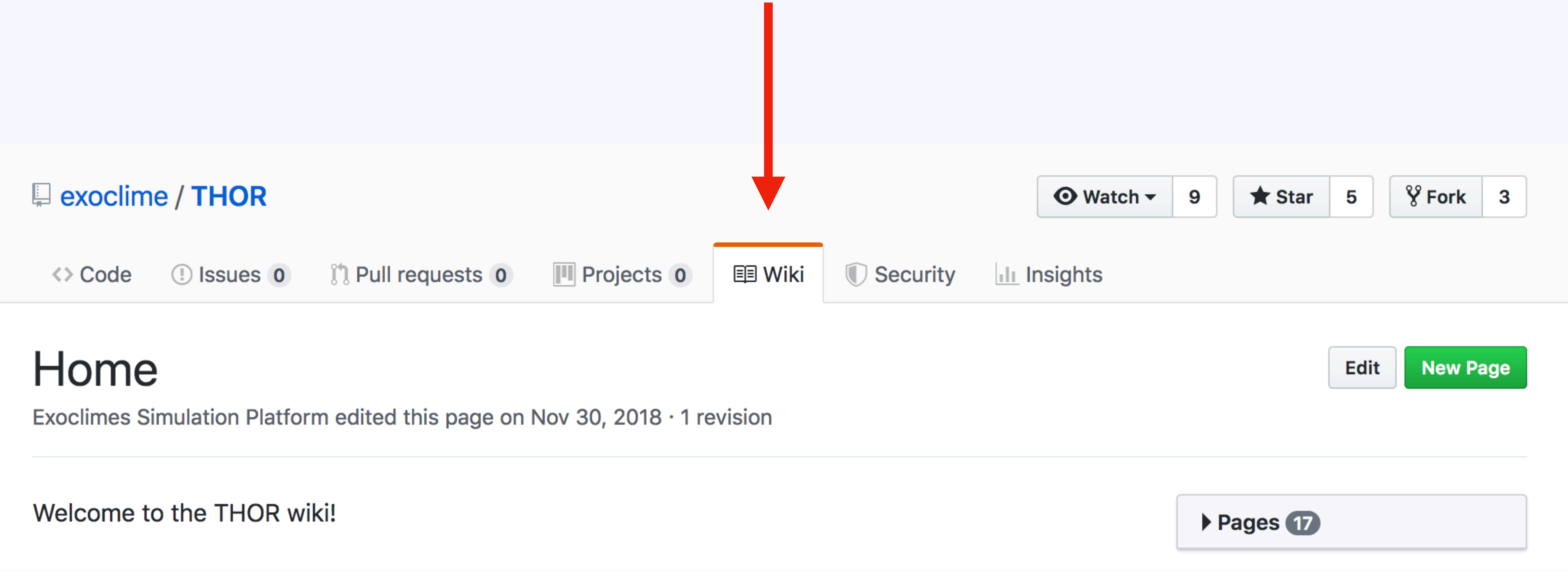
Using THOR

-adventures in general circulation modeling-

Russell Deitrick (with João Mendonça, Urs Schroffenegger, & Simon Grimm)

Wiki

- https://github.com/exoclime/THOR/wiki



Compiling!

- https://github.com/exoclime/THOR/wiki/Installing-and-compiling
- Dependencies: nvidia-cuda-toolkit, HDF5, make, git, gcc, g++, (plotting: python 3, h5py, pyshtools)
- Since we're running on Bern's GPU machine (Hulk), we'll bypass most of the steps (sorry)
  - `$ git clone https://github.com/exoclime/THOR.git`

Warning for Anaconda users: the make file for THOR will (try to) auto-detect the location of your hdf5 libraries. Unfortunately, Anaconda installs components of hdf5 which tend to interfere with the auto-detection process. Before compiling and running THOR, you may need to run `conda deactivate` (you can restart Anaconda with `conda activate`).

Compiling!

- One manual step
  - `$ cp Makefile.conf.template Makefile.conf`
  - Edit the new file (Makefile.conf):

```
deitrick@hulk:/storage/deitrick/THOR-dev$ more Makefile.conf
# Local makefile configuration
$(info Local Config)

MODULES_SRC := src/physics/managers/multi/
SM:=35
```

Connects physics modules

Compute capability of your GPU

- `$ make release -j8`

Compile optimized version

Parallel compile with 8 CPUs for speed (remove if you want compiler to print messages in order)

- When you run into compiler errors, first try:
  - `$ make clean` (then recompile as above)

Setting up your planet

- Let's look at an input file (https://github.com/exoclime/THOR/blob/master/ifile/earth_hstest.thr)
- Use nano or emacs (or vi/vim) on Hulk

```
# config file for THOR
# config format version
config_version = 1

# earth held-suarez model (Held & Suarez 1994)


#-- Time stepping and output options -----------------------------------------#
# number of steps
num_steps = 103680

# length of timesteps in seconds
timestep = 1000

# output
# output every n steps
n_out = 1920

```

= comment

Running the model

- https://github.com/exoclime/THOR/wiki/Running-THOR
- Running locally (direct access to GPU)
  - `$ bin/esp ifile/myplanet.thr`
- Running with Slurm scheduler (cluster-style)
  - `$ sbatch myjobscript`

```
#!/bin/bash
#SBATCH -D /home/thoruser/THOR/
#SBATCH -J earth
#SBATCH -n 1 --gres gpu:1
#SBATCH --time 0-2
#SBATCH --mail-type=ALL
#SBATCH --mail-user=thoruser@thormail.com
#SBATCH --output="/home/thoruser/slurm-esp-%j.out"

srun bin/esp ifile/config.thr
```

Running the model

- Flags:

```
-g / --gpu_id <N>            GPU_ID to run on
-o / --output_dir <PATH>     directory to write results to
-i / --initial <PATH>        initial conditions HDF5 filename
-N / --numsteps <N>          number of steps to run
-w / --overwrite             Force overwrite of output file if they exist
-c / --continue <PATH>       continue simulation from this output file
-b / --batch                 Run as batch
```

Local GPU
- `$ bin/esp ifile/myplanet.thr -w`

Hulk
- `$ srun --gres=gpu:1 bin/esp ifile/myplanet.thr -w`
  - Overwrite existing data!!

Local GPU
- `$ bin/esp ifile/myplanet.thr -b`

Hulk
- `$ srun --gres=gpu:1 bin/esp ifile/myplanet.thr -b`
  - "batch" mode: if **no** data exists, start from beginning; if data exists, start from last save file (see esp_log_write_<planet>.txt)

Output!

Data about performance

Global quantities (energy, etc.)

Log file (stdout)

```
deitrick@hulk:/scratch/deitrick/THOR-dev$ ls earth_hs/
esp_diagnostics_Earth.txt   esp_output_Earth_20.h5   esp_output_Earth_35.h5   esp_output_Earth_5.h5
esp_global_Earth.txt        esp_output_Earth_21.h5   esp_output_Earth_36.h5   esp_output_Earth_50.h5
esp_log_Earth.log           esp_output_Earth_22.h5   esp_output_Earth_37.h5   esp_output_Earth_51.h5
esp_output_Earth_0.h5       esp_output_Earth_23.h5   esp_output_Earth_38.h5   esp_output_Earth_52.h5
esp_output_Earth_1.h5       esp_output_Earth_24.h5   esp_output_Earth_39.h5   esp_output_Earth_53.h5
esp_output_Earth_10.h5      esp_output_Earth_25.h5   esp_output_Earth_4.h5    esp_output_Earth_54.h5
esp_output_Earth_11.h5      esp_output_Earth_26.h5   esp_output_Earth_40.h5   esp_output_Earth_6.h5
esp_output_Earth_12.h5      esp_output_Earth_27.h5   esp_output_Earth_41.h5   esp_output_Earth_7.h5
esp_output_Earth_13.h5      esp_output_Earth_28.h5   esp_output_Earth_42.h5   esp_output_Earth_8.h5
esp_output_Earth_14.h5      esp_output_Earth_29.h5   esp_output_Earth_43.h5   esp_output_Earth_9.h5
esp_output_Earth_15.h5      esp_output_Earth_3.h5    esp_output_Earth_44.h5   esp_output_grid_Earth.h5
esp_output_Earth_16.h5      esp_output_Earth_30.h5   esp_output_Earth_45.h5   esp_output_planet_Earth.h5
esp_output_Earth_17.h5      esp_output_Earth_31.h5   esp_output_Earth_46.h5   esp_write_log_Earth.txt
esp_output_Earth_18.h5      esp_output_Earth_32.h5   esp_output_Earth_47.h5   figures
esp_output_Earth_19.h5      esp_output_Earth_33.h5   esp_output_Earth_48.h5
esp_output_Earth_2.h5       esp_output_Earth_34.h5   esp_output_Earth_49.h5
```

Grid info and user settings

Simulation data at output time #

Note that these data are instantaneous, but I plan to add averages over the output interval. Be mindful of the tradeoff between output cadence and data size!

Log of output files

Some knobs you can turn

```
# Reference surface pressure [Pa]
P_ref = 10000000.0


#-- Grid options ---------------------------------------------------------------#
# Altitude of the top of the model domain [m]
Top_altitude = 1.4e6

# Horizontal resolution level.
glevel = 4

# Number of vertical layers
vlevel = 40

# Spring dynamics
spring_dynamics = true

# Parameter beta for spring dynamics
spring_beta = 1.15

```

Pressure at bottom of model

Model top (more on this later)

Grid points

$$N = 2 + 10 \times 2^{2 g_{\mathrm{level}}}$$

Vertical levels

(I never mess with these)

The "native" state of the model is NonHydrostatic, Deep

```
#-- Model options ---------------------------------------------------------------#
# Non-hydrostatic parameter
NonHydro = true

# Deep atmosphere
DeepModel = true
```

You can switch these off to experiment with hydrostatic and shallow approximations (warning: model usually does not perform as well)

Other things to be aware of

```
#-- Device options ------------------------------------------------------------#
# GPU ID number
GPU_ID_N = 0
```

When running *locally*, when multiple GPUs are present

You can start from an output file by setting rest = false:

```
# Initial conditions
rest = true

# initial conditions file, used if rest is set to false
# (path relative to current working directory)
# defaults to 'ifile/esp_initial.h5'
initial = ifile/esp_initial.h5

```

"ifile/esp_initial_planet.h5" must also be present

Forcing the atmosphere

- Benchmarks (Newtonian cooling)

```
# Core benchmark tests
# Held-Suarez test for Earth == HeldSuarez
# Benchmark test for shallow hot Jupiter == ShallowHotJupiter
# Benchmark test for deep hot Jupiter == DeepHotJupiter
# Benchmark test for tidally locked Earth == TidallyLockedEarth
# No benchmark test == NoBenchmark (model is then forced with grey RT by default)
core_benchmark = HeldSuarez
```

- Radiative transfer (double gray)

```
#-- Radiative transfer options (core_benchmark = 0) --------------------------#
## RT parameters #######################################
radiative_transfer = true
```

- Benchmark refs: Held & Suarez 1994, Cooper & Showman 2005, 2006, Menou & Rauscher 2009, Merlis & Schneider 2010, Rauscher & Menou 2010, Heng et al. 2011, Mayne et al. 2014, Mendonça et al. 2016

- RT refs: Lacis & Oinas 1991, Frierson et al. 2006, Heng et al. 2011, Mendonça et al. 2018

RT parameters (double gray)

- Incident stellar flux

$$S_0 = \sigma T_\star^4 (R_\star / a)^2 (1 - \alpha)$$

```
# stellar temperature (k)
Tstar = 4300

# orbital distance or semi-major axis (au)
planet_star_dist = 0.015

# radius of host star (R_sun)
radius_star = 0.667

# bond albedo of planet
albedo = 0.18
```

(yes, probably more options than necessary…)

- Internal heat flux (currently only with no surface)

```
# temperature of internal heat flux (bottom boundary) (K)
Tlow = 970
```

If lowest layer temperature is greater than this, then it is ignored…

RT parameters (double gray)

- Angle "integration" $dE(\mu) = B(\tau') \exp\left(\frac{-\tau'}{\mu}\right) \frac{d\tau'}{\mu}$ $\mu = \cos\theta$
  - Currently, for gray approx, we use diffusivity factor with $B(\tau')$ given by Stefan-Boltzmann (flux, instead of intensity)

```
# diffusivity factor
diff_ang = 0.5
```

$\mu = 1/\mathcal{D}$ $1 < \mathcal{D} < 2$

(bad naming choice, I know)

- Surface properties (RT)

```
# include surface heating
surface = false
# heat capacity of surface
Csurf = 1e7
```

RT parameters (double gray)

- Optical depths

$$\tau_{\rm sw} = \tau_{\rm sw,0}\left(\frac{P}{P_{\rm ref}}\right)^{n_{\rm sw}} \qquad \tau_{\rm lw} = f_{\rm lw}\tau_{\rm lw,0}\left(\frac{P}{P_{\rm ref}}\right) + (1-f_{\rm lw})\tau_{\rm lw,0}\left(\frac{P}{P_{\rm ref}}\right)^{n_{\rm lw}}$$

n = 1 (uniformly mixed absorber)
n > 1 (stronger in lower atmosphere)

mixed unmixed

```

# grey opt. depth of thermal wavelengths (at ref pressure)
taulw = 2128

# grey opt. depth of incoming stellar flux (at ref pressure)
tausw = 532
```

```

# power law index of unmixed absorbers (lw and sw)
n_lw = 2
n_sw = 1
# strength of unmixed absorbers in lw
f_lw = 0.5
```

- Special tuning for Earth (you can generalize this, if you feel like it)

```

# add sin(lat)^2 dependence to tau lw (advanced)
latf_lw = false
# opt depth at poles (lw)
taulw_pole = 1.5

```

$$\tau_{\rm lw,0} = \tau_{\rm lw,eq} + (\tau_{\rm lw,pole} - \tau_{\rm lw,eq})\sin^2\phi$$

Insolation

```
## insolation (orbit + spin-state) parameters ###########
# synchronous rotation (tidally-locking at 1:1)
sync_rot = true

# mean motion of orbit (if sync_rot=false and ecc>0) (rad/s)
#mean_motion = 1.98e-7

# initial substellar longitude (deg)
#alpha_i = 0

# initial orbital position (deg)
#true_long_i = 0

# eccentricity of orbit
#ecc = 0

# obliquity (axial-tilt) (deg)
#obliquity = 0

# longitude of periastron (relative to equinox) (deg)
# (stupid Earth convention applies)
#longp = 0
#########################################################
```

Ensures substellar longitude does not drift (overrides mean_motion)

Coded for arbitrary rotation/orbit

Still testing, so please let me know if you use these parameters and what you learn! (esp. if you find mistakes)

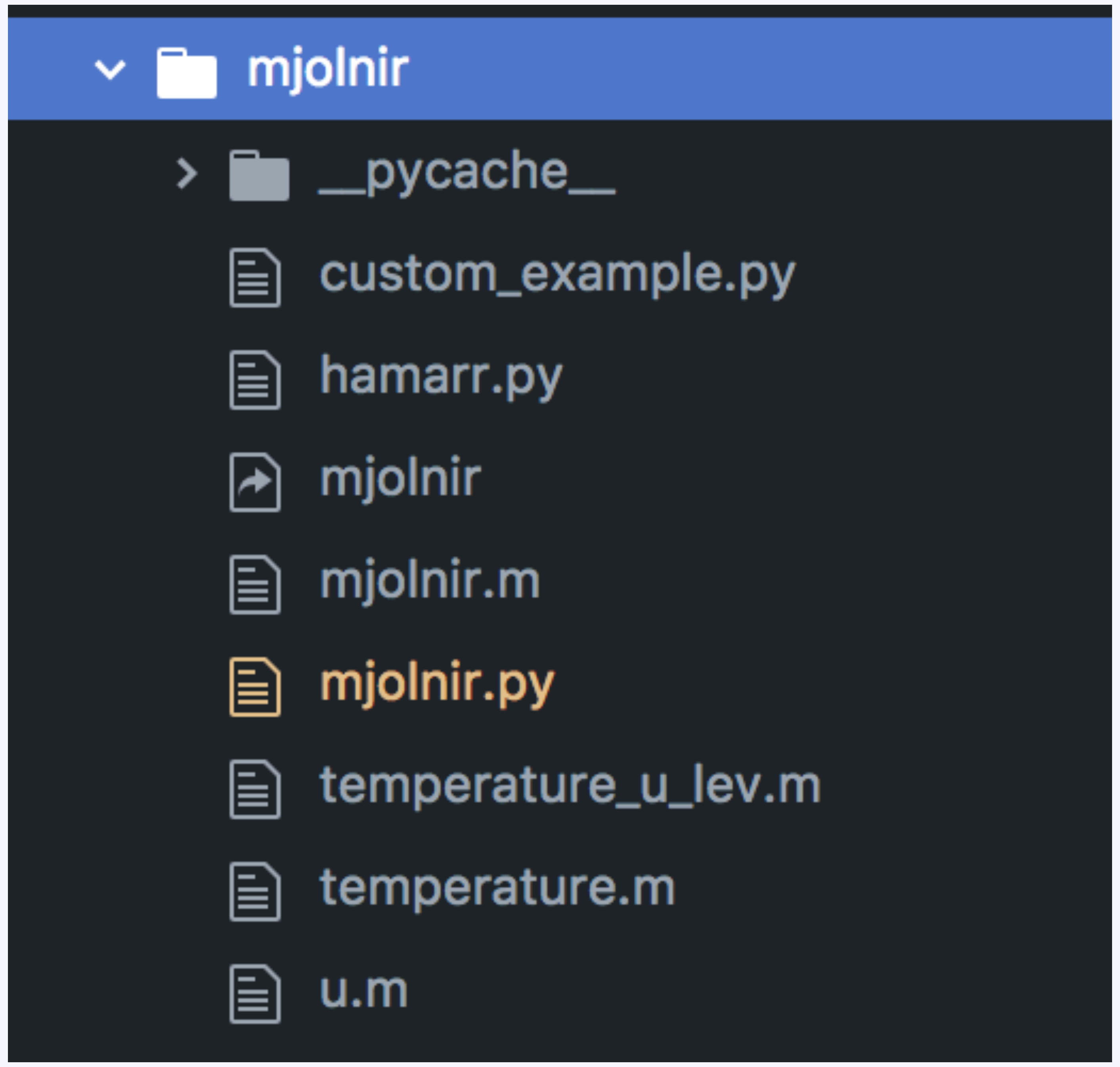
```
# Rotation rate [rad s^-1]
rotation_rate = 9.09E-5
```

This can be negative! (retrograde spin)

Making plots

- To copy output files to another computer (I like “rsync”):
    - `$ rsync -vr user@hulk.unibe.ch:<path_to_output> <local_path>`
- See https://github.com/exoclime/THOR/wiki/Python-plotting
- An additional dependency: pyshtools (https://pypi.org/project/pyshtools/)
    - `$ pip3 install pyshtools`
    - Used only for calculating KE spectra—if you have trouble installing pyshtools, you can comment out the code that uses it…

Making plots



The Python code is written by me (based on João's MATLAB code) and is changing all the time, so be sure to commit changes you make to it!

"mjolnir.py" is the main Python script which import "hamarr.py"

"custom_example.py" shows you how to customize some things

The MATLAB code is written by João and is pretty old, but it is there if you like MATLAB and want something to get started with

Note: please view these as a starting point. You will probably need to make changes and write your own scripts to plot new things. There may also be mistakes, so please don't use without understanding what the scripts do

Making plots

You can run the Python code on the command line (you'll need to update your PATH and PYTHONPATH first):

```
deitrick@Deitricks-MacBook:~/Code/THOR-dev$ mjolnir -i 60 -f earth_rt_new_nh_dconv Tver -pmin 10
/Users/deitrick/Code/THOR-dev/mjolnir/mjolnir:8: DeprecationWarning: the imp module is deprecated i
n favour of importlib; see the module's documentation for alternative uses
  from imp import reload
/Users/deitrick/Code/THOR-dev/mjolnir/hamarr.py:926: UserWarning: No contour levels were found with
in the data range.
  c2 = ax.contour(latp*180/np.pi,rg.Pressure[prange[0],0]/1e5,Zonallt[:,prange[0]].T,levels=levp,co
lors='w',linewidths=1)
0.9022867679595947
```

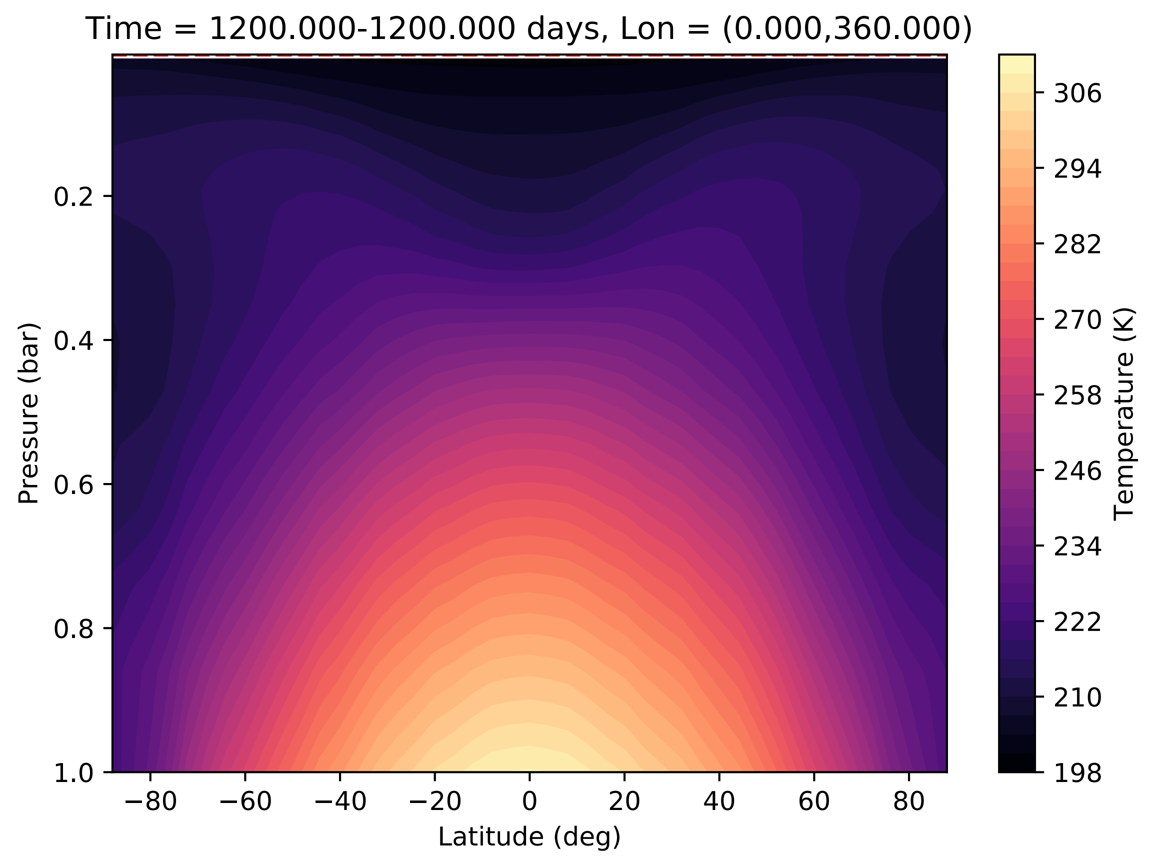
(Oops, looks like I need to update something…)

```
$ mjolnir -i <start> -l <end> -f <results folder> <plot type>
```

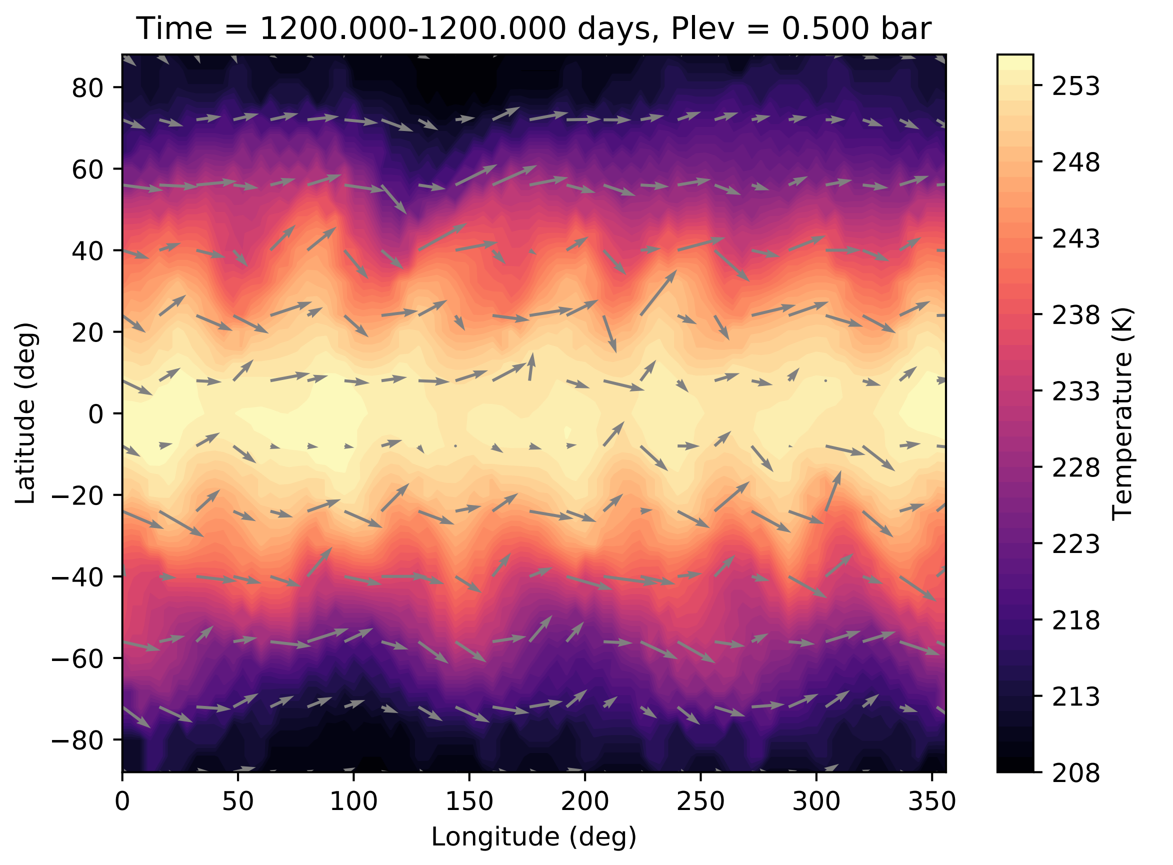
first and last output
file numbers

Making plots

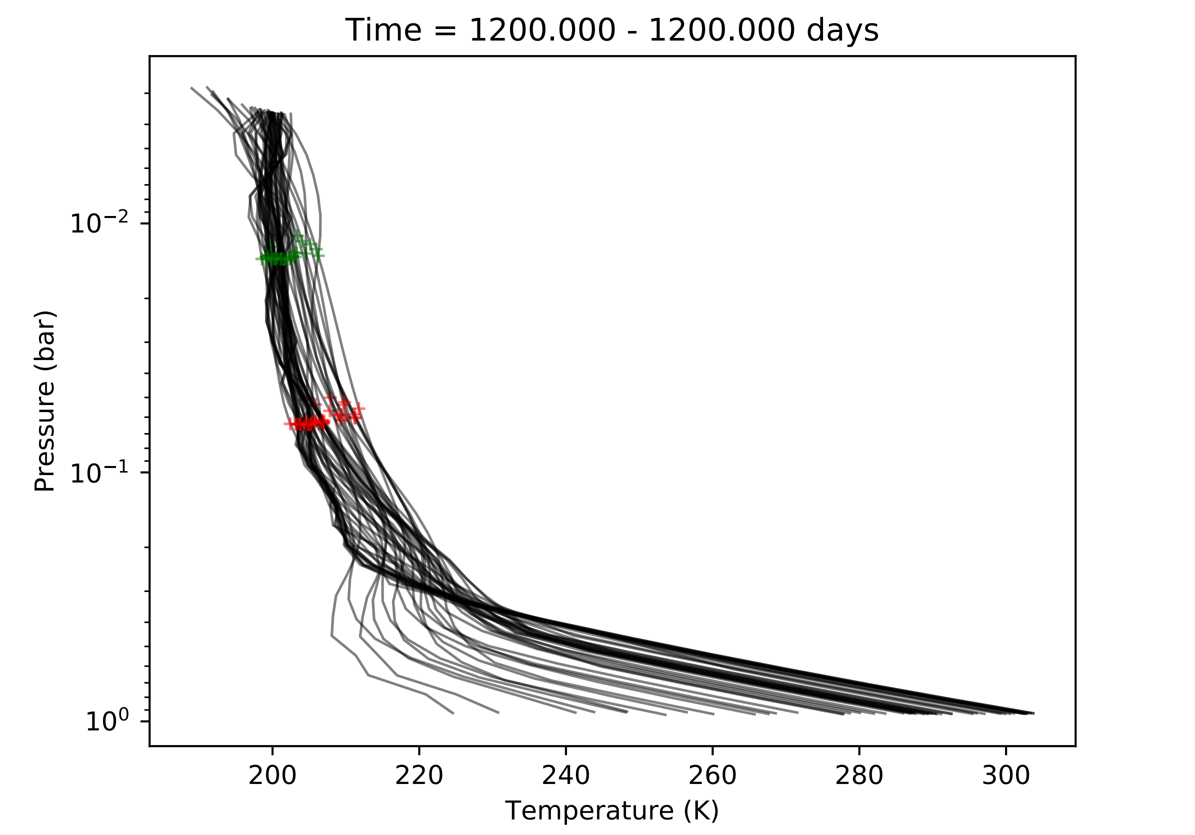
Plot types: vertical, horizontal, profile, others…



Vertical



Horizontal



Profile

```
valid = ['uver','wver','wprof','Tver','Tulev','PTver','ulev','PVver','PVlev',
            'TP','RVlev','cons','stream','pause','tracer','PTP','regrid','KE',
            'SR','uprof','cfl']
```

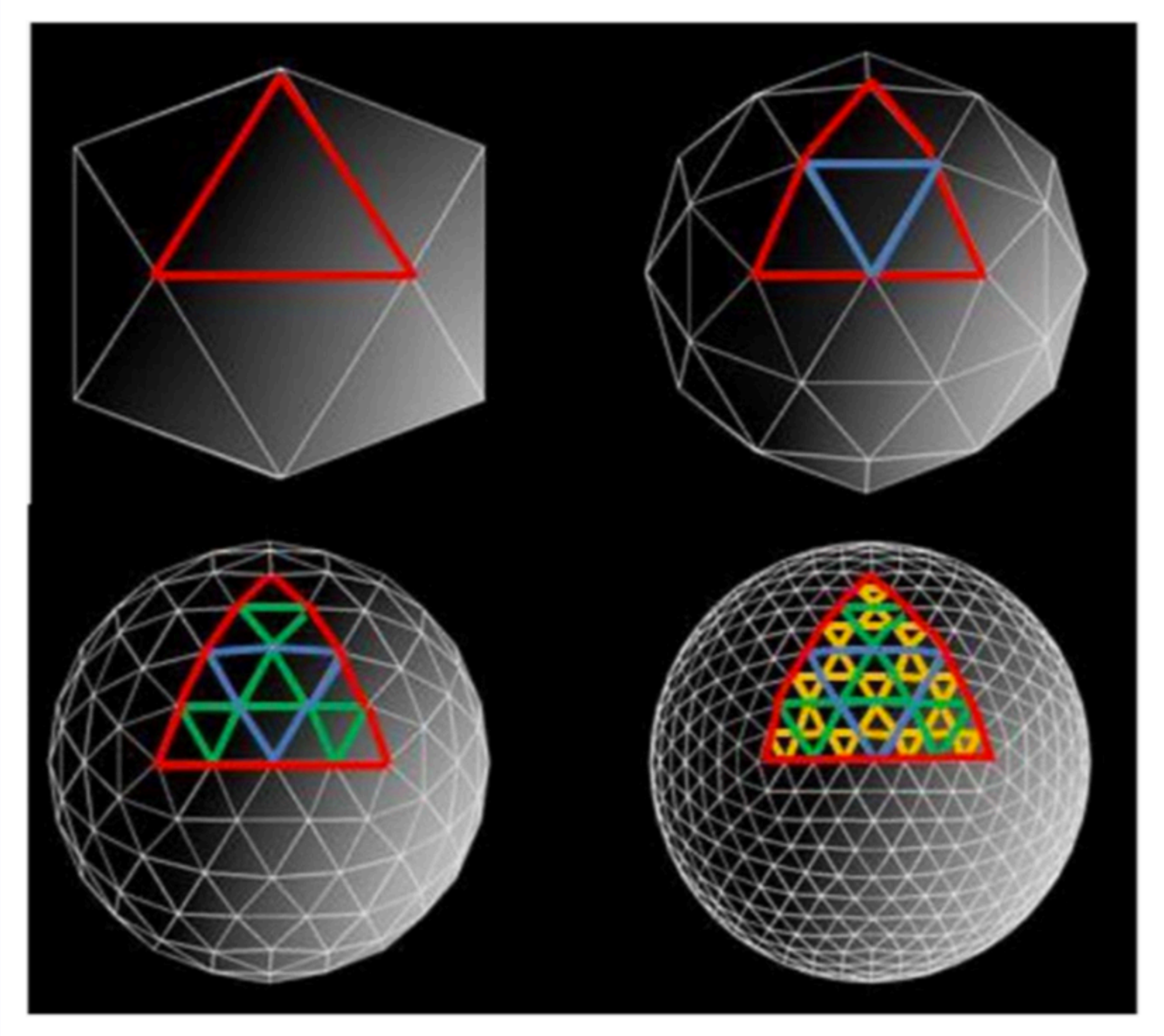
I'm always adding to these…

Making plots

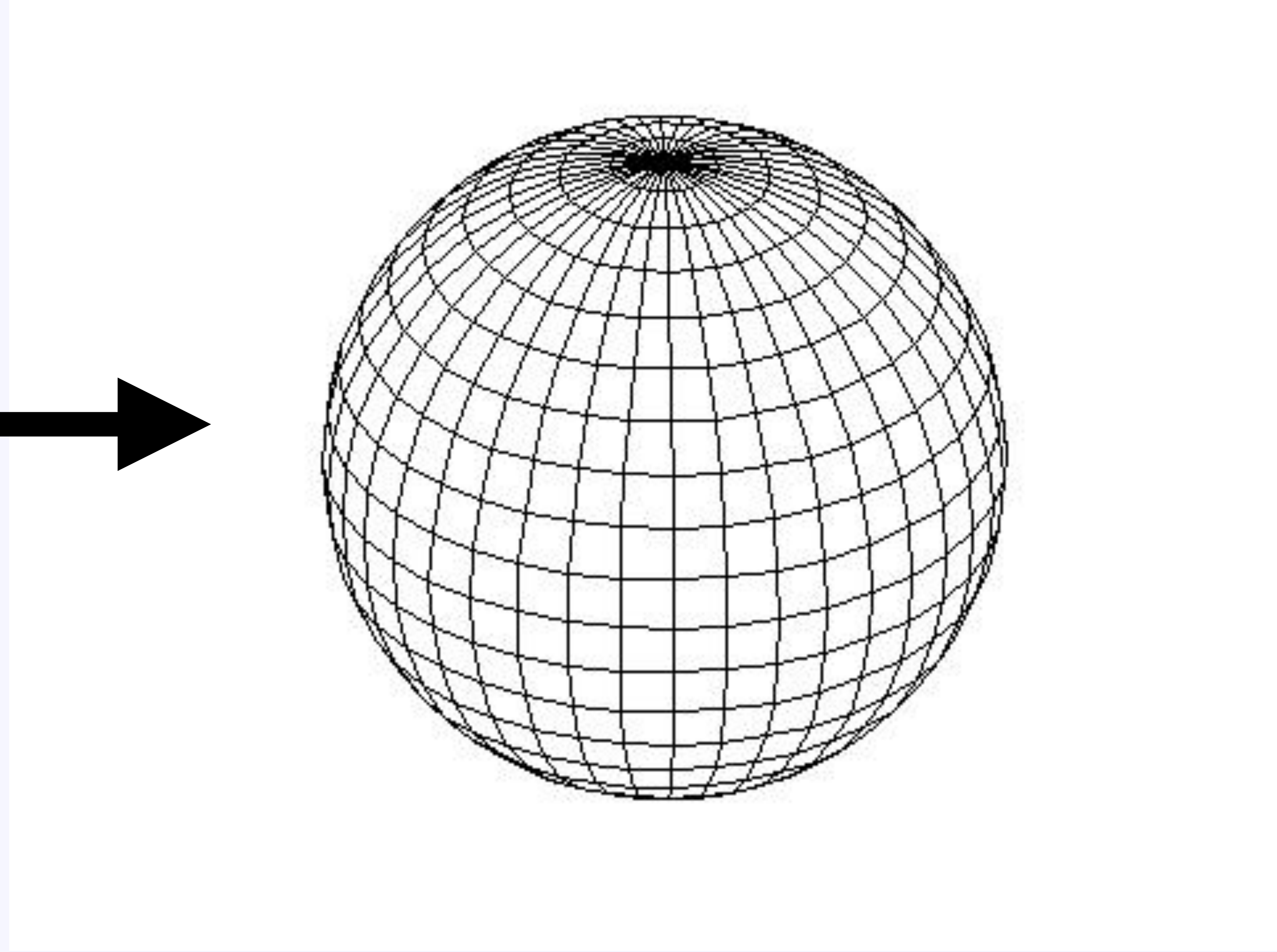
“regrid” is a special mjolnir argument

```
$ mjolnir -i <start> -l <end> -f <results folder> regrid
```

icosahedral/height



latitude/longitude/pressure



Word of caution:
I use a 2-D flattened interpolation function from SciPy for the horizontal regridding. This produces artifacts at the edges (long =0, 360) and poles. Probably acceptable for plots but could be improved on!

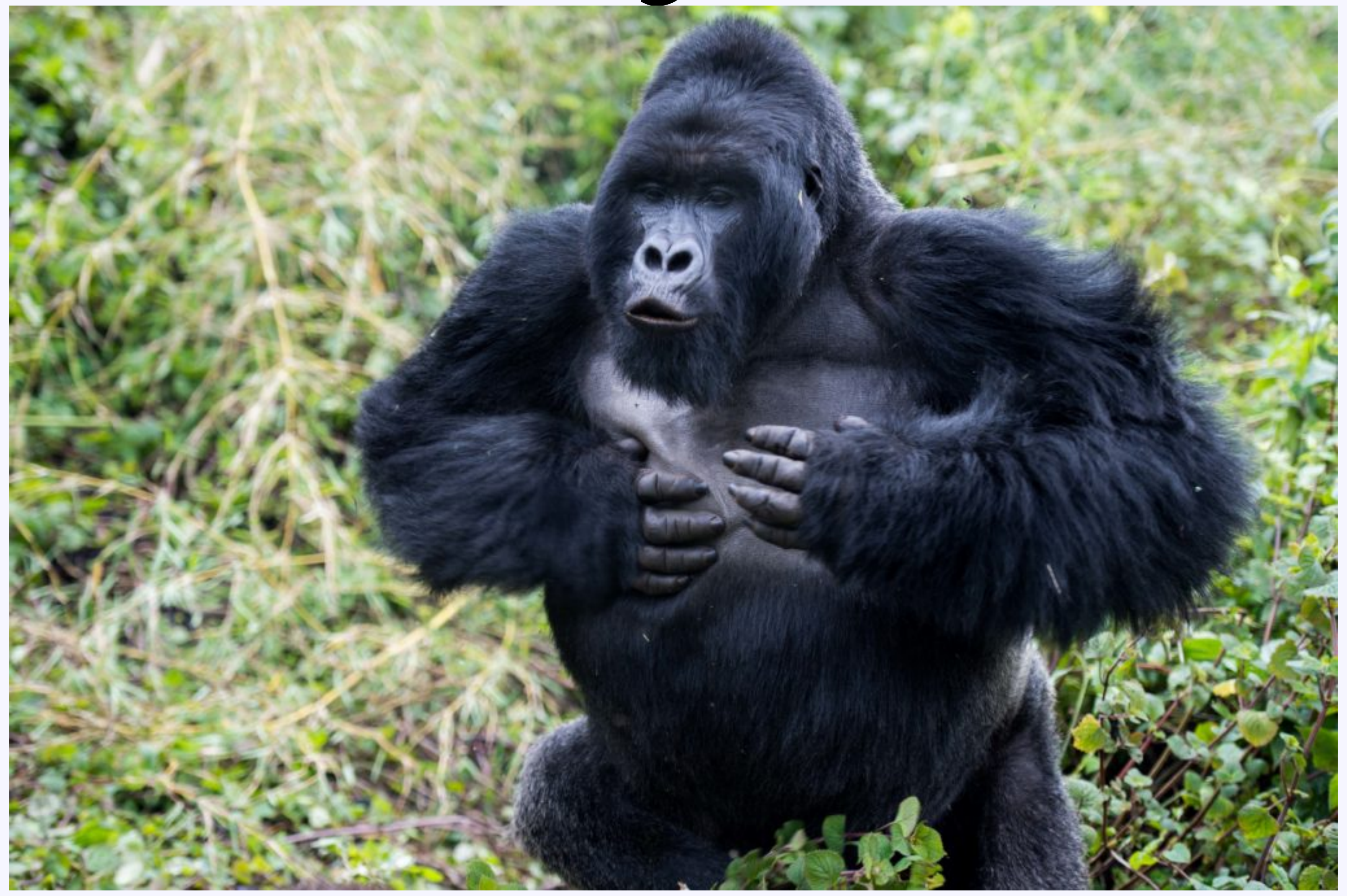
All contour plots require the lat-lon-pressure grid. Suggestion: run regrid on all simulation files overnight! Then plotting is much faster.

How do I stabilize the model??

- Get coffee
- Open input file and tweak, run, tweak, run, tweak, run…
- Make plots (lots and lots of plots)

Model stability

- Hyperdiffusion
- Divergence damping
- Time step
- Sponge layer (hot planets)
- Things to explore: boundary conditions, initial conditions, vertical diffusion, temperature sponge, other numerical tricks??



a.k.a., wrestling with the 400 pound gorilla

Hyperdiffusion & divergence damping

$$
\begin{aligned}
F_\rho &= \boxed{-\nabla_h^2 K_{\text{hyp}} \nabla_h^2 \rho} \\
F_{\vec{v}_h} &= -\nabla_h^2 \rho K_{\text{hyp}} \nabla_h^2 \vec{v}_h \boxed{- K_{\text{div}} \nabla_h^2 \nabla (\nabla \cdot (\rho \vec{v}))} \\
F_{v_r} &= -\nabla_h^2 \rho K_{\text{hyp}} \nabla_h^2 v_r \\
F_P &= -R_d \nabla_h^2 \rho K_{\text{hyp}} \nabla_h^2 T
\end{aligned}
$$

"divergence damping":
unique to split time-stepping algorithm with "fast" and "slow" terms

"hyperdiffusion":
standard damping for GCMs

∇_h^2 = horizontal Laplace operator

Shamrock & Klemp 1992, Tomita & Satoh 2004, Mendonça et al. 2016

Hyperdiffusion & divergence damping

$$
\begin{aligned}
F_\rho &= -\nabla_h^2 K_{\text{hyp}} \nabla_h^2 \rho \\
F_{\vec{v}_h} &= -\nabla_h^2 \rho K_{\text{hyp}} \nabla_h^2 \vec{v}_h - K_{\text{div}} \nabla_h^2 \nabla (\nabla \cdot (\rho \vec{v})) \\
F_{v_r} &= -\nabla_h^2 \rho K_{\text{hyp}} \nabla_h^2 v_r \\
F_P &= -R_d \nabla_h^2 \rho K_{\text{hyp}} \nabla_h^2 T
\end{aligned}
$$

```
## diffusion ##############################################
# Hyper-diffusion
HyDiff  =    true

# Divergence-damping
DivDampP =    true

# Strength of diffusion
Diffc = 0.015

# Strength of divergence damping
DivDampc = 0.015
##########################################################
```

D

$$K = D \frac{d^4}{\Delta t}$$

$$d = \sqrt{\frac{2\pi}{5}} \frac{r_0}{2^{g_{\text{level}}}}$$

Damping time scale:

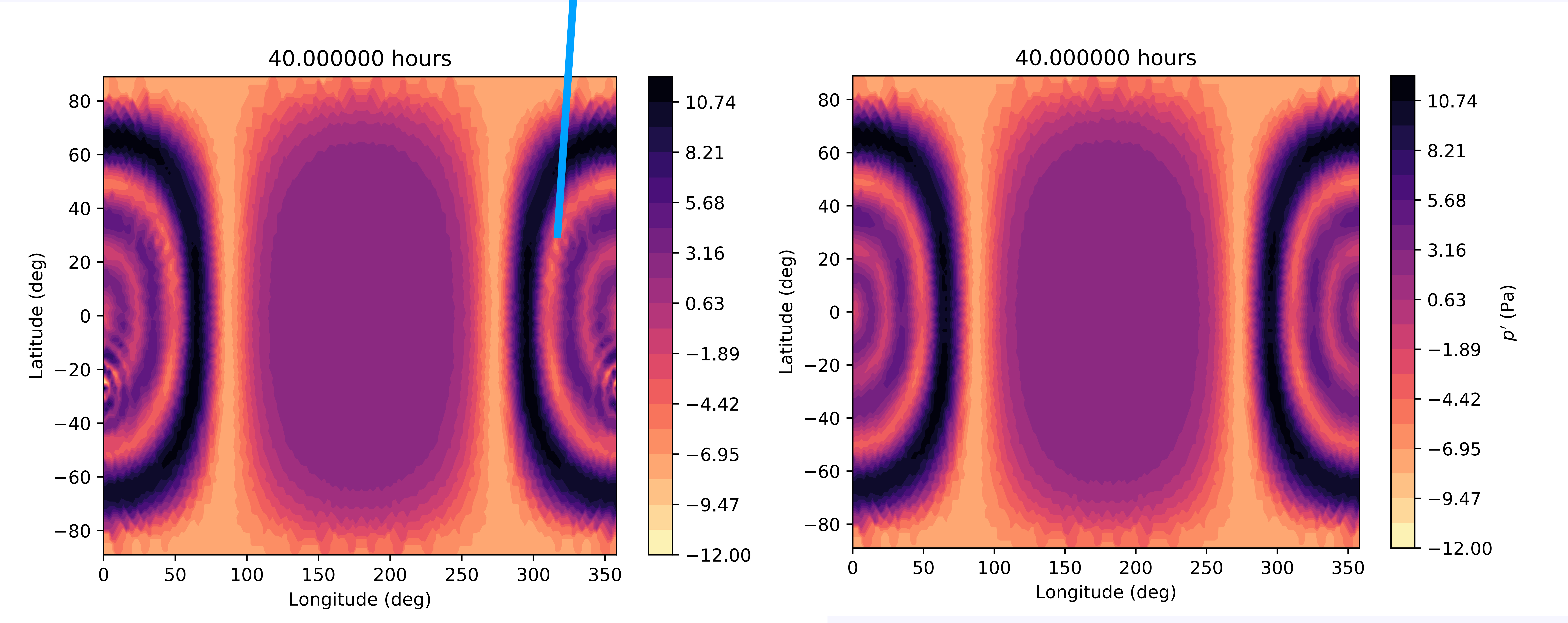
$$\tau_d = \frac{d^4}{2^5 K} = \frac{\Delta t}{2^5 D}$$

The need for damping

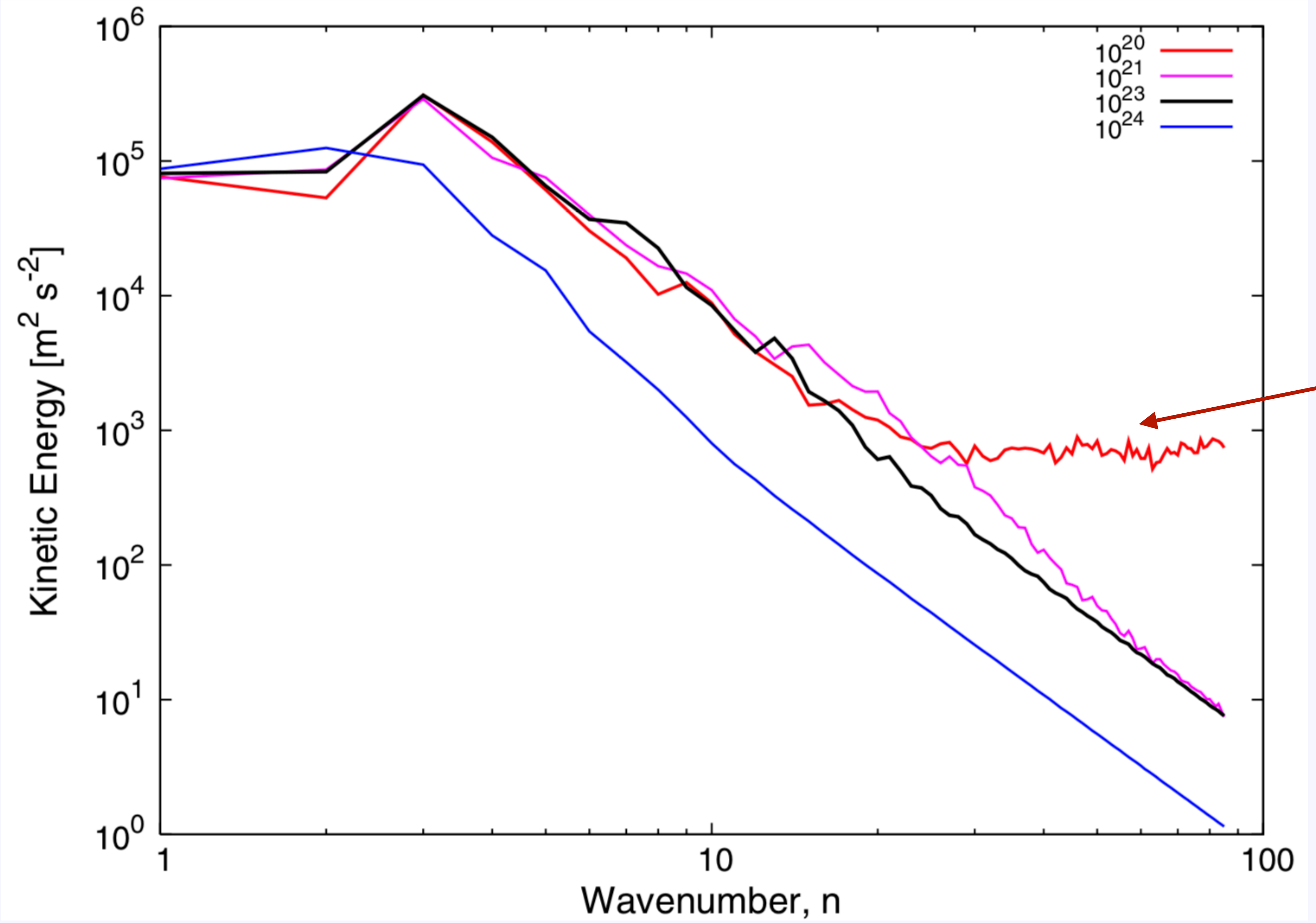
No damping

Pentagons!

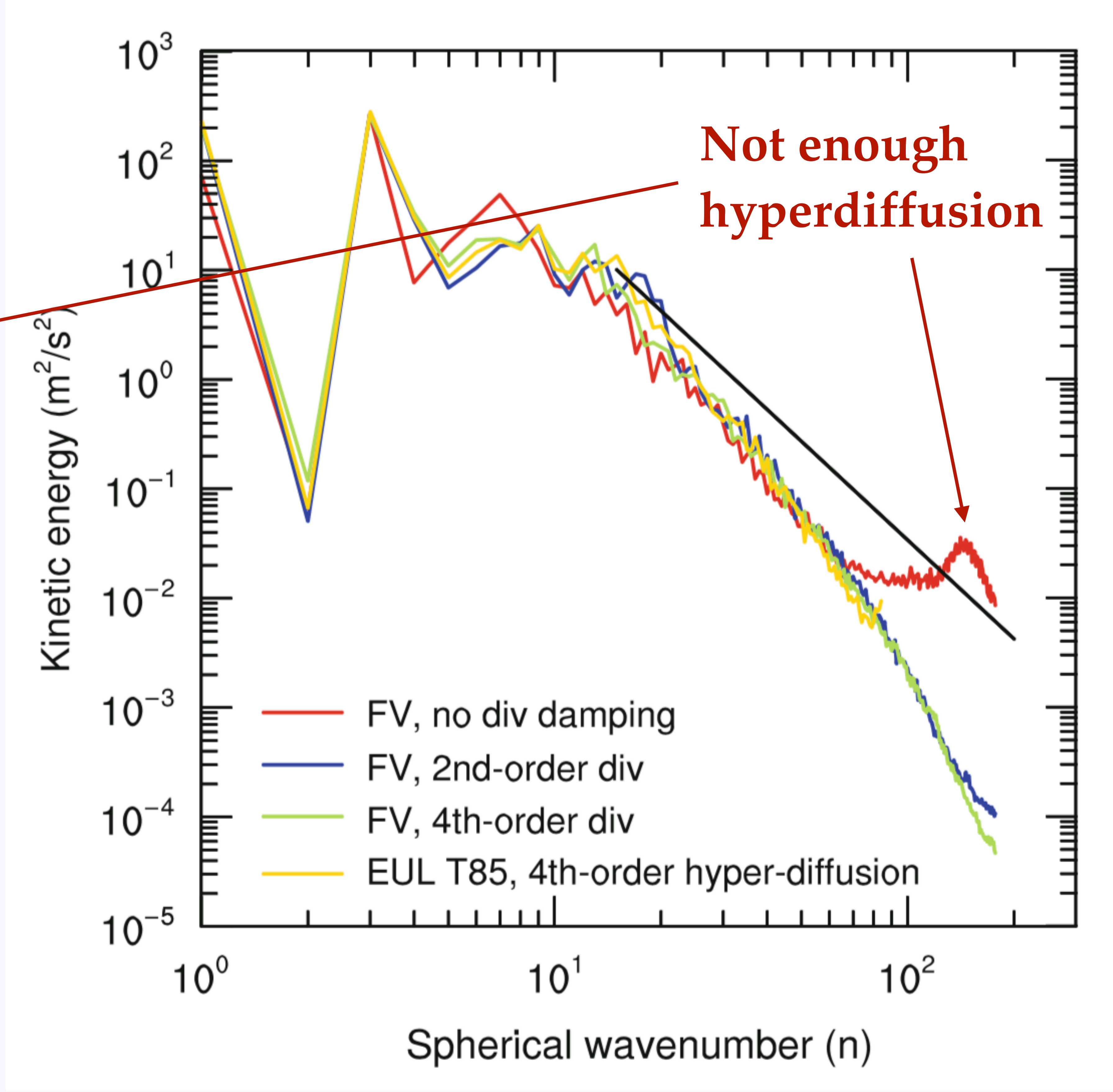
Divergence damping



The need for damping



Thrastarson & Cho 2011



Jablonowski & Williamson 2011
(in *Numerical Techniques for Global Atmospheric Models*)

Energy cascade (turbulence) brings energy to small scale. In a real atmosphere, this dissipates because of molecular viscosity, but GCMs usually don't resolve this…

How do I pick a time step?

- Trial & error (mostly)
- Guiding principles
  - CFL number

$$C = \frac{u\Delta t}{\Delta x} < 1$$

Try: sound speed and horizontal grid size

$$c_s = \sqrt{\gamma R_d T} \qquad d = \sqrt{\frac{2\pi}{5}\frac{r_0}{2^{g_{\mathrm{level}}}}}$$

(yes, we have acoustic waves in THOR)

  - Radiative time scale (Showman & Guillot 2002)

$$\tau_{rad} \sim \frac{C_P P}{4\sigma g T^3}$$

Usually shortest at top of model

How do I pick a time step?

- Time steps can be too *small*
  - Partly, this is due to diffusion scaling, which can lead to overdamping:

$$K = D\frac{d^4}{\Delta t}$$

  - Partly, a mystery we are still trying to solve
    - One hypothesis is that we begin to resolve faster waves but resolve them poorly (João disagrees…)

Adjust boundaries (model top, especially)

- We solve the fluid equations on a height grid, rather than a pressure grid

- So we have to *solve* for pressure from density and potential temperature:

$$\frac{\partial \rho\theta}{\partial t} + \nabla \cdot (\rho\theta\vec{v}) = 0 \qquad\qquad P = P_{\mathrm{ref}}\left(\frac{R_d\ \rho\theta}{P_{\mathrm{ref}}}\right)^{C_P/C_V}$$

from Mendonça et al 2016

- The numerics at low pressure are fickle—sometimes you can get negative potential temperature or pressure (then the model crashes)

- Pressures ~< 10^{-5} bar are usually where things get sketchy, so be mindful where you set the model top!

- But beware: making the top too low can produce weird results!

Sponge layer (for hot planets)

- Hot atmospheres tend to have strong vertically p
(gravity) waves

- The best way to conserve mass, energy, etc., is to apply the boundary conditions at top and bottom:

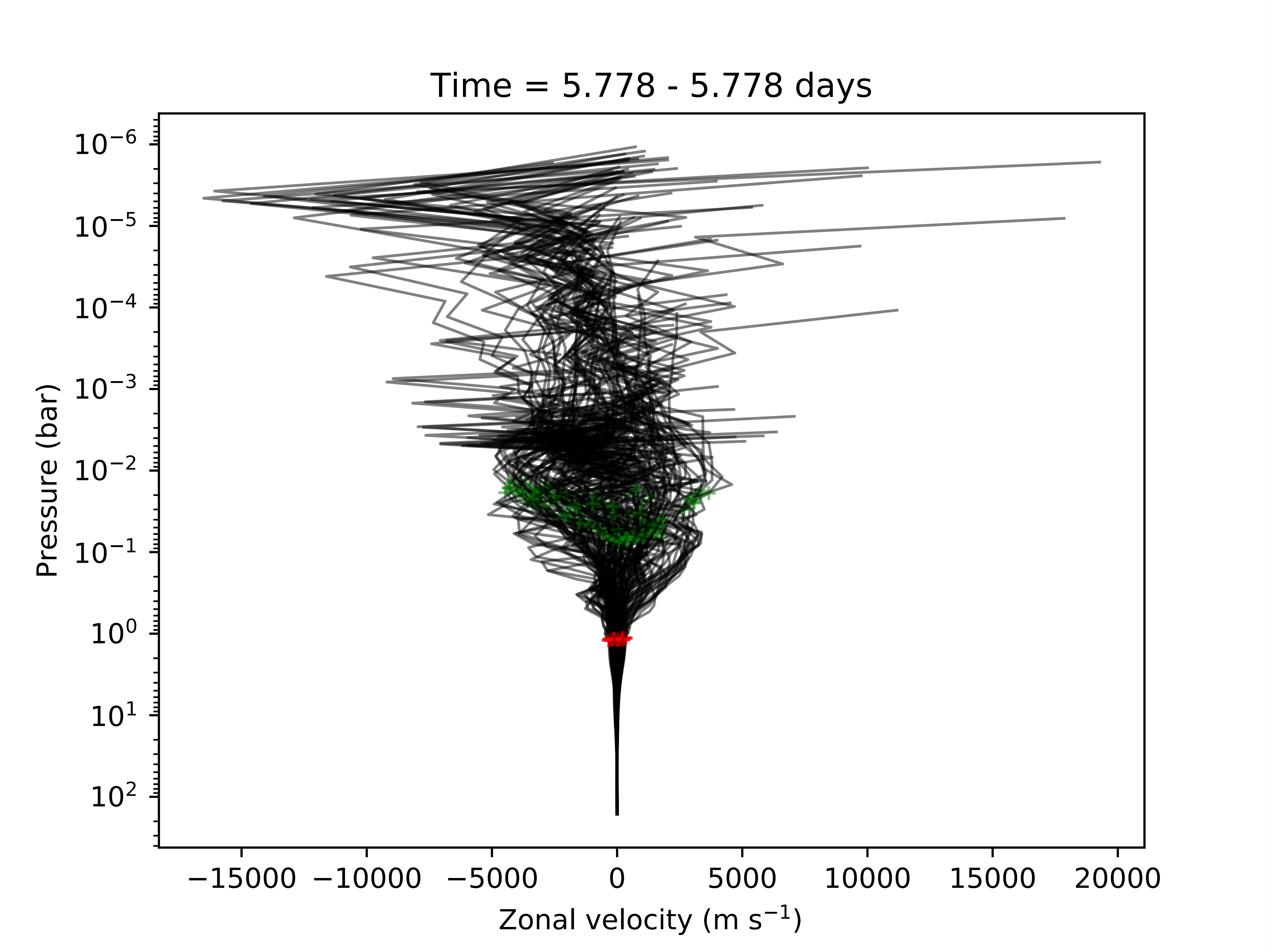
$$v_r = 0$$

- (this is a *reflective* boundary)

- (so waves that should continue up and dissipate at pressure we can't hope to model, instead bounce back and can constructively interfere)

The result…

- Steep gradients (which are bad news for numerical solvers)



Sponge layer (for hot planets)

$$\frac{dv}{dt} = -\frac{v - \langle v \rangle}{\tau}$$

damp velocities toward zonal mean


```
#-- Sponge layer (Rayleigh drag) ---------------------------------------------#
# use sponge layer (Rayleigh drag) at top of atmosphere?
SpongeLayer = true

# latitude rings (zonal mean is calculated over these)
nlat = 20

# bottom of sponge layer (fractional height)
ns_sponge = 0.75

# strength of sponge layer (1/damping time)
Rv_sponge = 1e-4

# shrink sponge by half after some time (experimental)
#shrink_sponge = true

# when to shrink sponge (days)
#t_shrink = 1
```

bins used to calculate zonal mean

height where sponge layer begins

strength of damping (1 / time scale)

You can dissipate the sponge (strength decays exponentially). The sponge can also be removed manually after some time by stopping, editing the config file, and restarting, but this is not as smooth.

I changed this to # of time steps (haven't updated all the config files, sorry)

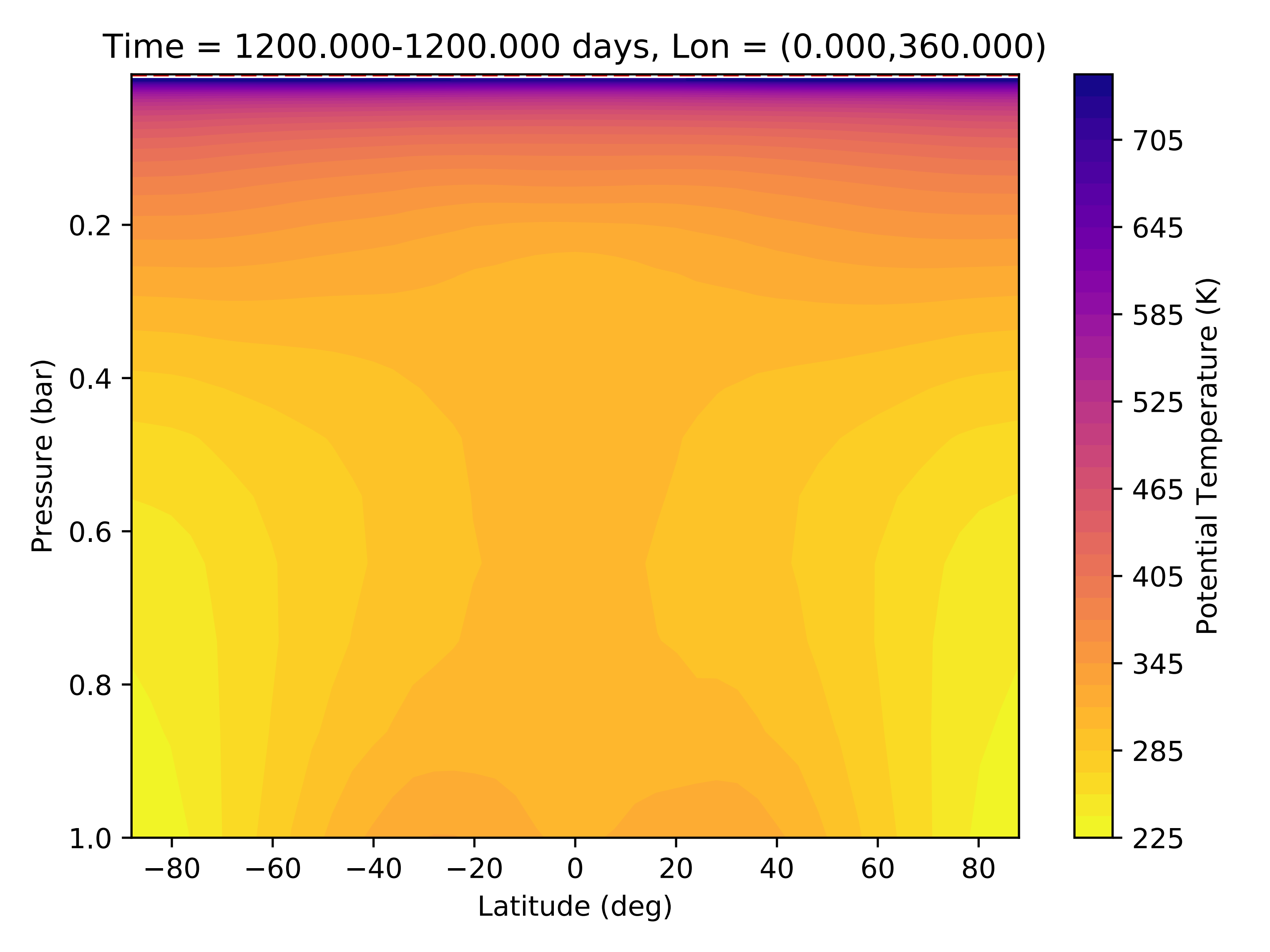
Dry convective adjustment

Manabe et al., 1965

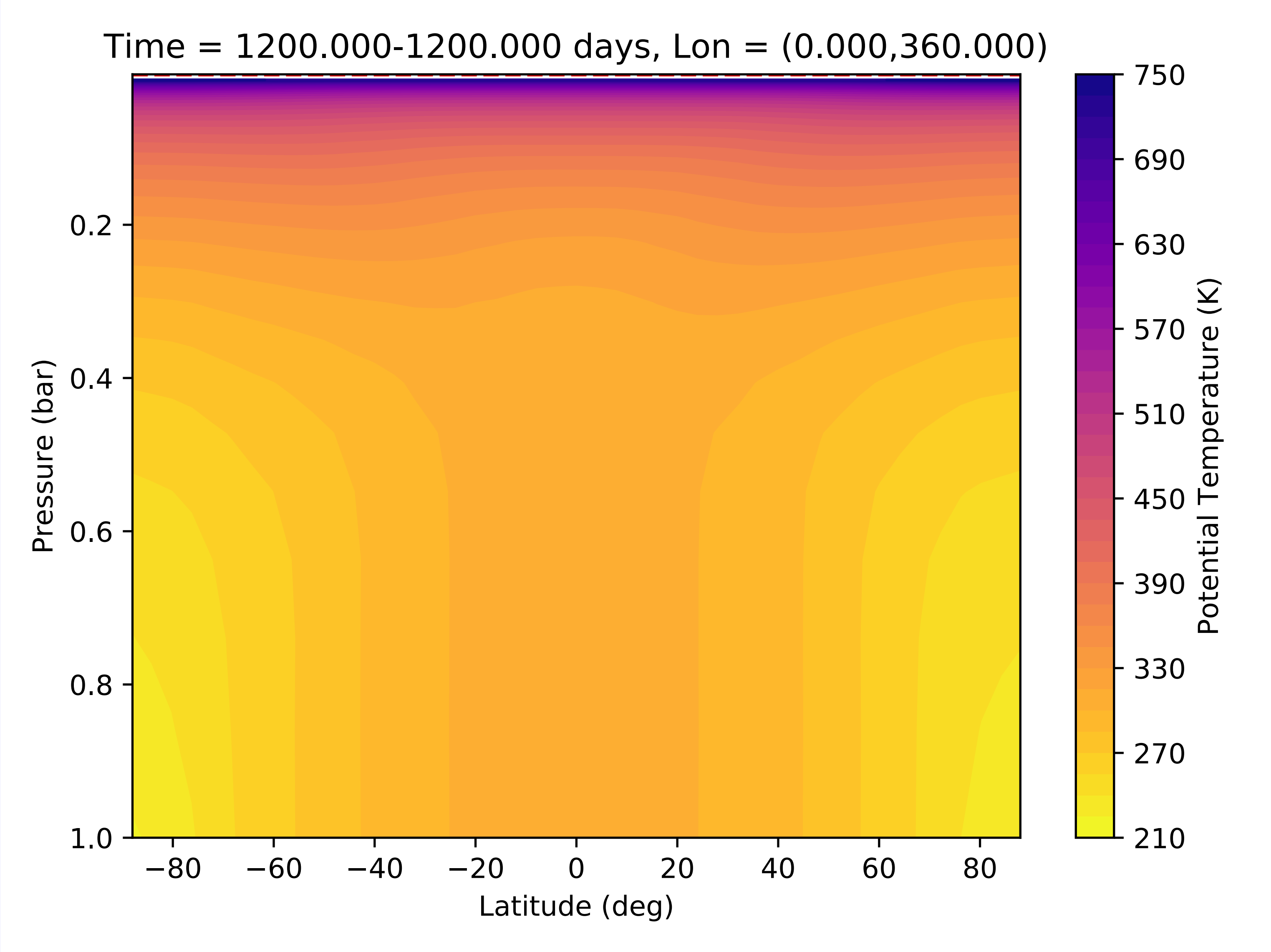
```
# use convective adjustment scheme
conv_adj = 1
```

(sorry about the inconsistent usage of true/false vs 1/0)

Earth sim without conv_adj



Earth sim with conv_adj



(secret: this can help stabilize the numerics in some situations)

Advanced debugging

- You can compile in debug mode and run cuda-gdb to interact with the code directly as it runs (must have direct access to GPU):
  - `$ make -j8 debug`
  - `$ cuda-gdb bin/esp`
- Then you can set break points and such (very similar to standard gdb)

Advanced debugging

- Check out src/headers/debug.h (for serious coders)

```
// benchmarking
// if defined run benchmark functions?
// #define BENCHMARKING

// ***************************************
// * binary comparison
// compare benchmark point to references
// #define BENCH_POINT_COMPARE
// write reference benchmark point
// #define BENCH_POINT_WRITE
// print out more debug info, by default, only print out failures
// #define BENCH_PRINT_DEBUG
// print out comparisaon statistics
//#define BENCH_COMPARE_PRINT_STATISTICS
// use an epsilon value for fuzzy compare on relative value
// #define BENCH_COMPARE_USE_EPSILON
// #define BENCH_COMPARE_EPSILON_VALUE 1e-7
// ***************************************
// * check for NaNs
// #define BENCH_NAN_CHECK
// * below adds checks on device functions (useful for device memory bugs)
// #define BENCH_CHECK_LAST_CUDA_ERROR

```

Uncomment to turn on code "benchmarking" before compile

Options for comparing run to run

Give more info about where/why code crashed!

Contribute to THOR

- Fork the repo
- Add some code/fix some bugs/improve the code
- Submit a pull request!

Test!

- Copy the directory below to your own THOR directory

```
deitrick@hulk:~/THOR/summer_2019_trials$ pwd
/home/deitrick/THOR/summer_2019_trials
deitrick@hulk:~/THOR/summer_2019_trials$ ls
deephj_2019.thr  earth_acoustic_test_2019.thr  wasp43b_1_2019.thr  wasp43b_2_2019.thr
```

- Each of the simulations will crash! Can you make them run for the entire num_steps?
    - (Clues are in the names of the output directories)
    - (I won't guarantee they are stable after num_steps…)