IN THIS MODULE:

• Overview of python plotting routines
BEFORE WE BEGIN:

• Change to the premade sample_run directory

• Copy python files to sample_run

$ cp ../rayleigh/analysis_routines/python/* .py .

• Check directory contents: $ ls

• You should see a lot of .py files
## FILES YOU SHOULD HAVE

<table>
<thead>
<tr>
<th>FILENAME</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>diagnostic_reading.py</td>
<td>Reading routines for each output</td>
</tr>
<tr>
<td>plot_azavg.py</td>
<td>AZ_Avg plotting example</td>
</tr>
<tr>
<td>plot_energy_distro.py</td>
<td>plots radial variation of KE</td>
</tr>
<tr>
<td>plot_energy_flux.py</td>
<td>plots radial energy flux balance</td>
</tr>
<tr>
<td>plot_energy_trace.py</td>
<td>plots time-evolution of energy</td>
</tr>
<tr>
<td>plot_reference.py</td>
<td>plots the reference state</td>
</tr>
<tr>
<td>plot_shell_slice.py</td>
<td>Shell_Slice plotting example</td>
</tr>
<tr>
<td>plot_spectrum.py</td>
<td>plots power spectrum</td>
</tr>
</tbody>
</table>
EXAMPLE/EXERCISE: ENERGY VS. TIME

$ python plot_energy_trace.py

Each plotting routine opens up an interactive window.
PLOTTING Routines: Overview

• Diagnostics are written in unformatted binary

• Each diagnostic type has an associated python class

• Each has a corresponding reading routine in: diagnostic_reading.py

• Various plotting routines are provided

• All routines contain certain similar elements
$ gedit plot_energy_trace.py

# Global-Averages (G Avgs) plotting example
# Reads in time steps 2.7 million through 4 million
# Plots average KE vs time
#
# This example routine makes use of the GlobalAverage
# data structure associated with the G_Avg output.
# Upon initializing a GlobalAverage object, the
# object will contain the following attributes:
#
# self.niter : number of time steps
# self.nq : number of diagnostic quantities output
# self.qv[0:nq-1] : quantity codes for the diagnostics output
# self.vals[0:niter-1,0:nq-1] : Globally averaged diagnostics as function of time and diagnostic
# self.iters[0:niter-1] : The time step numbers stored in this output file
# self.time[0:niter-1] : The simulation time corresponding to each time step
# self.lut : Lookup table for the different diagnostics output

Summary of relevant diagnostic class
BRIEF NOTE: DIAGNOSTIC CLASSES

- Each diagnostic class has this minimal set of attributes
- Actual data is always contained in “vals” numpy array
- Additional attributes for more complex diagnostic classes
- The dimension of vals depends on the diagnostic type
COMMON ELEMENT #2:
THE LOOKUP TABLE (LUT)

myfile = ‘G_Avgs/00000010’
g = GlobalAverage(filename=myfile,path=‘’)

ke_index = a.lut[125]  # kinetic energy location
me_index = g.lut[475]  # magnetic energy location

ke = g.vals[:,ke_index]
me = g.vals[:,me_index]

• KE, ME probably not located at indices 125, 475 of vals...
• Use the lookup table to find them
• Lookup table is run dependent!
COMMON ELEMENT #3: BUILD_FILE_LIST

```python
files = build_file_list(100, 10000, path='G_Avgs')
```

- Simple python routine that finds all files within specified directory and time-step range.
- This example finds filenames between: 00000100 and 000010000.
- Help average or concatenate data from multiple files
COMMON ELEMENT #4: SAVEPLOT & SAVEFILE

- All plotting routines can be run interactively by setting `saveplot = False` (default)
- Setting `saveplot=True` will forgo interactive window and save to indicated file
- You may also save in interactive mode using disk icon
- Let’s run through the remaining examples...

```python
saveplot = False
savefile = 'energy_trace.pdf'
```
EXAMPLE: SHELL AVERAGES

$ python plot_energy_distro.py

NOTE: Shell_Avgs output also contains variance, skewness, and kurtosis
EXAMPLE: SHELL AVERAGES

$python plot_energy_flux.py

![Graph of energy flux vs radius showing various components: total, cond, heat, enth, visc, KE.](image)
EXAMPLE: AZIMUTHAL AVERAGES

$ python plot_azavg.py
Rayleigh creates a file named “reference” at runtime
EXAMPLE: SHELL SLICES

$ python plot_shell_slice.py
EXAMPLE: SHELL SPECTRA

$ python plot_spectrum.py
Exercise 1:

- Return to the module5 directory
- Create a local copy of the python routines
- Modify plot_shell_slice.py to display the three components of $\mathbf{B}$ at three depths
Exercise 2:

- Create a directory named module6
- Copy c2001_case0_input to main_input
- Set res to $nr = 64$ and $ntheta = 48$
- Output global averages every 50 time steps
- Run for 20,000 timesteps
- Modify plot_energy_trace.py to display KE vs. time only