RAYLEIGH TUTORIAL

MODULE 2: DIAGNOSTIC OUTPUTS

BEFORE WE BEGIN:

- Create a directory named module5
- Copy c2001_case0_input to module5/main_input
- DELETE benchmark_mode = 1
- TURN Magnetism ON and magnetic_init_type = 7
- Set to run for 50 time steps
- Set n_theta = 48
- Softlink rayleigh to module5 directory

IN THIS MODULE:

- Overview of Diagnostic Mechanics
- Diagnostic Types
- Diagnostic Quantities & Menu System
- Plotting Output

RAYLEIGH DIAGNOSTICS: OVERVIEW

- Rayleigh performs a number of in-situ diagnostics
- Helps to reduce disk usage and post-processing
- In-situ diagnostics represent varying degrees of slicing and averaging
- Controlled via the output namelist
- Each diagnostic stored in dedicated directory
- All outputs performed via MPI-IO (no HDF5 or NetCDF)
- Format is independent of processor layout
- Can be read into Python using provided routines

RAYLEIGH DIAGNOSTICS: MECHANICS

- Examine the output_namelist in main_input
- The prefix indicates the diagnostic TYPE
- This is a particular TYPE of analysis that MAY be performed
- Output TYPE examples:
  - a slice (e.g. equatorial plane or meridional plane)
  - an average (e.g., full-volume average or average in phi)
  - a full 3-D data cube
- We will cover all available output TYPES soon…

```
&output_namelist
TYPE_values  = 1, 2, 3, 64
TYPE_frequency = 100
TYPE_nrec = 10
```

RAYLEIGH DIAGNOSTICS: MECHANICS

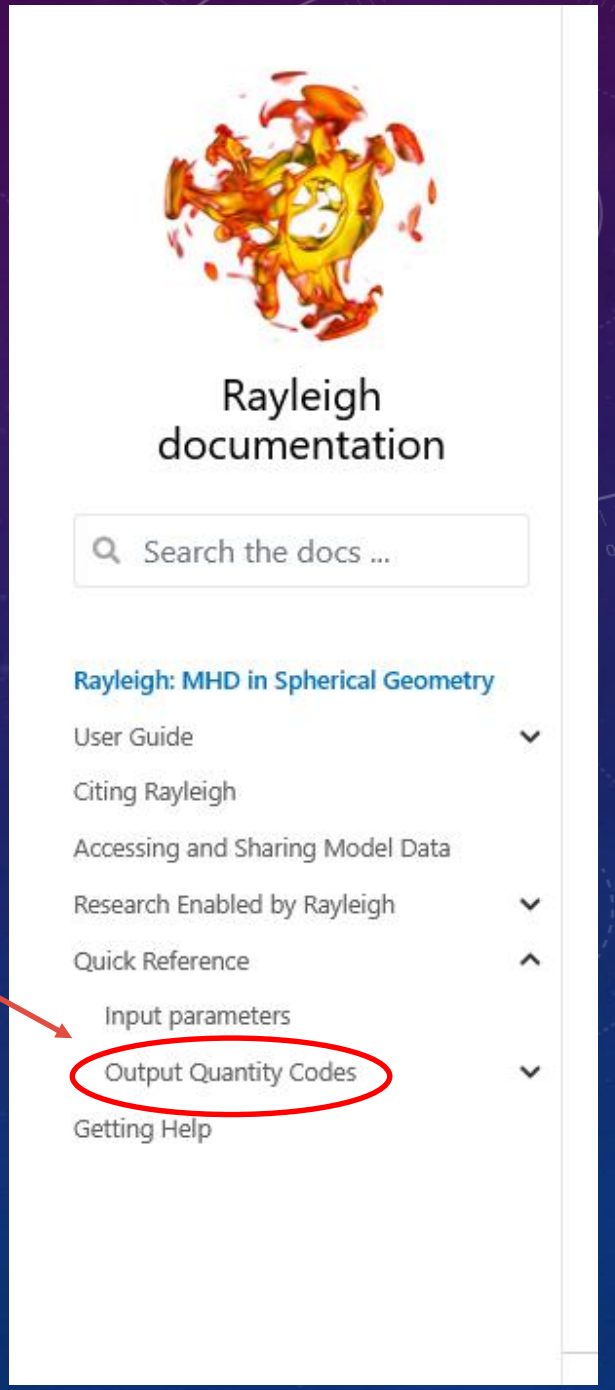
- VALUES indicates WHAT should be analyzed in TYPE fashion
- Numbers are code for physical quantities selected from menu
- In this example, we are analyzing the three velocity components (1,2,3) and temperature or entropy (501)
- If no values are specified, this TYPE of output is not performed

```
&output_namelist
TYPE_values  = 1, 2, 3, 501
TYPE_frequency = 100
TYPE_nrec = 10
```

RAYLEIGH DIAGNOSTICS: Quantity Codes

- Rayleigh has a large number of possible output quantities (e.g., velocity, temperature)
- Each has a unique associated numeric output code
- Enumerated in the documentation
  - Most frequently visited page in docs…

- Note that each output quantity is computed only once and then passed to the slicing and averaging routines
  - No significant penalty for outputting same quantity in multiple ways



RAYLEIGH DIAGNOSTICS: MECHANICS

- FREQUENCY indicates HOW OFTEN we perform THIS analysis
- Units are in time steps
- In this example, we perform our analysis of velocity and temperature once every 100 time steps

```
&output_namelist
TYPE_values  = 1, 2, 3, 501
TYPE_frequency = 100
TYPE_nrec = 10
```

RAYLEIGH DIAGNOSTICS: MECHANICS

- NREC indicates HOW MANY time steps worth of analyses are saved to one file

- In this example, 10 analyses are saved within each file.

- We generate a new file every 1000 time steps (nrec*freq)

```
&output_namelist
TYPE_values  = 1, 2, 3, 501
TYPE_frequency = 100
TYPE_nrec = 10
```

EXERCISE:

- Before we go any further, let's try this out
- Modify these portions (only) of main_input
- Run your code

```
&output_namelist
  globalavg_frequency = 2
  globalavg_nrec = 10

  shellavg_frequency = 5
  shellavg_nrec = 5
```

- Globalavg diagnostics are stored in G_Avgs
- Examine the contents of that directory
- Recall that
  - We output every 2nd time step
  - We stored 10 records per file

00000020 contains time steps:
2, 4, 6, 8, 10, 12, 14, 16, 18, 20

```
nick@nick-VirtualBox ~/Desktop/Rayleigh_Tutorial/module5 $ ls -lh G_Avgs/
total 12K
-rw-r--r-- 1 nick nick 472 Jun 17 14:57 00000020
-rw-r--r-- 1 nick nick 472 Jun 17 14:57 00000040
-rw-r--r-- 1 nick nick 252 Jun 17 14:57 00000060
```

00000060 contains time steps: 42, 44, 46, 48, 50

- Shellavg diagnostics are stored in Shell_Avgs
- Examine the contents of that directory
- Recall that
  - We output every 5th time step
  - We stored 5 records per file

00000025 contains time steps:
5, 10, 15, 20, 25

```
nick@nick-VirtualBox ~/Desktop/Rayleigh_Tutorial/module5 $ ls -lh Shell_Avgs/
total 208K
-rw-r--r-- 1 nick nick 101K Jun 17 14:57 00000025
-rw-r--r-- 1 nick nick 101K Jun 17 14:57 00000050
```

00000050 contains time steps: 30, 35, 40, 45, 50

Specifying Output Coordinates

- Some outputs require additional information
- For example, slices on spherical shells require the user to indicate the radius of the desired shells:

```
&output_namelist
  shellslice_levels      = 3,16,32,48,62
```

- This line specifies the INDICES of radii at which spherical surfaces or spectra are taken
- Index 1 corresponds to the upper boundary
- In our example, index 64 corresponds to the lower boundary

Specifying Output Coordinates

- One can also specify relative coordinates (preferred):

```
&output_namelist
  shellslice_levels_nrm      = 0.1, 0.5, 0.8
```

- In this example, shellslices would be output at a radii that are 10%, 50% and 80% of the way from the lower boundary.
- Recommended method because “immune” to resolution changes.
- We will examine several examples of this shortly

DIAGNOSTIC TYPES:

- Let's now explore the different outputs that are available in Rayleigh
- Two data sets are provided for today's tutorial
  - /Rayleigh_tutorial/anelastic
  - /Rayleigh_tutorial/Boussinesq

- A variety of Python notebooks designed to examine these datasets are available on Github
- Let's clone the repository and have a look…

```
$ cd ~
$ git clone https://github.com/feathern/rayleigh_tutorial_2025.git
$ cd rayleigh_tutorial_2025
```