# Rayleigh Hackathon 2019 Preliminary report

## Introduction

- Timeline
- Participants and areas of interest

## Timeline

## Participants and areas of interest

## Resources

- Git Tutorial:

## Report on projects the participants worked on

- Reorganize the documentation to an automatic online build
  - Nick Featherstone, Maria Weber, Lorraine Hwang, Rene Gassmoeller

- Add an automatic regression tester
  - Rene Gassmoeller

- Documentation Updates
  - Lorraine Hwang

- Custom reference states and nondimensionalizations, with Jupyter notebook examples
  - Nicholas Nelson, Philipp Edelmann, Ryan Orvedahl, Lydia Korre, Loren Matilsky, Nick Featherstone, Mike Calkins

- Jupyter Notebooks for visualization with YT
  - Maria Weber, Nick Featherstone

- Parallel routine to interpolate spherically gridded volumetric Rayleigh output onto a Cartesian grid
  - Kyle Augustson, Maria Weber, Nick Featherstone

- Statistics about Rayleigh’s growth during the hackathon
Introduction

To further develop the spherical convection/dynamo code Rayleigh and to grow and foster its open source community, 15 user-developers of Rayleigh worked side-by-side over a 5 day period at the University of Colorado, Boulder in July 2019.

The Rayleigh community made significant progress along various lines of project development. Some efforts begun at the workshop remain in their initial stages. This includes a multi-month effort to revamp Rayleigh’s parallel I/O and a new format for Checkpointing.

Several code additions came to fruition during the workshop, such as the new generic input interface that was wrapped into the code. This interface and supporting Python routines allow a user to define arbitrary initial conditions and arbitrary boundary conditions. With this addition, the possibilities for Rayleigh’s boundary and initial conditions are no longer limited to a few pre-programmed options. Imposed magnetic fields, non-spherically-symmetric temperature perturbations, and fixed-flow boundary conditions are now all possible.

A separate major outcome of the workshop was the addition of a custom-reference state interface to the code. Through the use of supporting Python routines, a user is now completely free to alter the constant and non-constant coefficients in the PDEs solved by Rayleigh. This feature allows users to adopt alternative nondimensionalizations from those already provided by Rayleigh. It also allows custom thermodynamic background states to be used (previously only polytropes were available). Attendees developed a number of Jupyter notebooks providing examples of how to use this new interface.

Representatives from the Vapor and YT teams also attended the workshop and provided demonstrations of their visualization software. Following their presentations, workshop participants developed a set of Jupyter notebooks and supporting documentation that provide and describe example visualization workflows based around Rayleigh’s Spherical_3D output.

Finally, several improvements to the documentation were developed during the workshop. Perhaps most importantly, the documentation was moved into Sphinx format and is now hosted on Readthedocs. Updates to the documentation are automatically processed during each merge, and the Readthedocs page updates automatically.

During the course of the hackathon, almost every participant contributed source code to the project. Together, users and developers changed a total of more than 60,000 lines of files. Due to some cleanup the number of source code lines decreased over the hackathon, even though new functionality was added. Although this result has to be interpreted with care during the hackathon more commits were added (223) than the project had in total before the hackathon started (220). This stems likely from the merge of some large contributions that were developed over the time before the hackathon and that were merged during this window.
Below is the timeline and a log of the individual contributions. Many of these contributions are discussed in greater detail following the table of participants’ interests.

### Timeline

<table>
<thead>
<tr>
<th>Day</th>
<th>Scheduled items</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monday, 07/22</td>
<td>9:00A: Participant Introductions, Topic groups</td>
</tr>
<tr>
<td></td>
<td>10:00A - 11:15A: Git tutorial</td>
</tr>
<tr>
<td>Tuesday, 07/23</td>
<td>9:00A: Morning discussion</td>
</tr>
<tr>
<td></td>
<td>10:00A: Discussion about custom reference states</td>
</tr>
<tr>
<td>Wednesday, 07/24</td>
<td>9:00A: Morning discussion</td>
</tr>
<tr>
<td></td>
<td>1:00P: Checkpoint discussion</td>
</tr>
<tr>
<td>Thursday, 07/25</td>
<td>9:00A: Morning discussion</td>
</tr>
<tr>
<td></td>
<td>11:00A: Yt presentation</td>
</tr>
<tr>
<td></td>
<td>1:00P: Vapor3 presentation</td>
</tr>
<tr>
<td>Friday, 07/26</td>
<td>9:00A: Morning discussion</td>
</tr>
</tbody>
</table>

### Participants and areas of interest

<table>
<thead>
<tr>
<th>Name, affiliation, email</th>
<th>Goals and interests for this hackathon</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rene Gassmoeller, UC Davis, <a href="mailto:rene.gassmoeller@mailbox.org">rene.gassmoeller@mailbox.org</a></td>
<td>1. Help others with their goals</td>
</tr>
<tr>
<td></td>
<td>2. Review pull requests</td>
</tr>
<tr>
<td></td>
<td>3. Set up regression tester</td>
</tr>
<tr>
<td></td>
<td>4. Set up documentation structure</td>
</tr>
<tr>
<td>Lorraine Hwang, UC Davis, <a href="mailto:ljhwang@ucdavis.edu">ljhwang@ucdavis.edu</a></td>
<td>1. Logistics</td>
</tr>
<tr>
<td></td>
<td>2. Documentation</td>
</tr>
<tr>
<td>Nick Featherstone, CU Boulder, <a href="mailto:nicholas.featherstone@colorado.edu">nicholas.featherstone@colorado.edu</a></td>
<td>1. Introducing people to Rayleigh’s design</td>
</tr>
<tr>
<td></td>
<td>2. Helping others</td>
</tr>
<tr>
<td>Cian Wilson, Carnegie Institution DTM</td>
<td>1. Discuss generic input</td>
</tr>
<tr>
<td></td>
<td>2. Implement compositional field</td>
</tr>
<tr>
<td>Email</td>
<td>1. Documentation</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td><a href="mailto:cwilson@carnegiescience.edu">cwilson@carnegiescience.edu</a></td>
<td>Maria Weber University of Chicago <a href="mailto:maweber@uchicago.edu">maweber@uchicago.edu</a></td>
</tr>
<tr>
<td>Lydia Korre LASP, CU Boulder <a href="mailto:lydia.korre@lasp.colorado.edu">lydia.korre@lasp.colorado.edu</a></td>
<td>1. Custom reference state &amp; nondimensionalizations (anelastic, Boussinesq)</td>
</tr>
<tr>
<td>Philipp Edelmann Newcastle University, UK <a href="mailto:philipp.edelmann@ncl.ac.uk">philipp.edelmann@ncl.ac.uk</a></td>
<td>1. Custom reference state</td>
</tr>
<tr>
<td>Peter Driscoll Carnegie Institution DTM <a href="mailto:pdriscoll@carnegiescience.edu">pdriscoll@carnegiescience.edu</a></td>
<td>1. Discuss generic input</td>
</tr>
<tr>
<td>Nicholas Nelson California State University, Chico <a href="mailto:njnelson@csuchico.edu">njnelson@csuchico.edu</a></td>
<td>1. Custom reference states from MESA</td>
</tr>
<tr>
<td>Michael Calkins University of Colorado, Boulder <a href="mailto:michael.calkins@colorado.edu">michael.calkins@colorado.edu</a></td>
<td>1. Learn more about using GitHub with Rayleigh</td>
</tr>
</tbody>
</table>
Resources

Git Tutorial:

- The slides from the first day presentation: https://www.dropbox.com/s/5logqymei6b8885/Git-Github-introduction.pptx?dl=0
- Github workflow: https://guides.github.com/introduction/flow/
- Git tutorial: https://swcarpentry.github.io/git-novice/

1. Explain and set up Git:

2. Explain Github Workflow:
   b. Ensure forked repositories
   c. Ensure proper remotes

3. Walkthrough
   a. Create Branch
      i. ‘git checkout master’
      ii. ‘git pull upstream master’
      iii. ‘git checkout -b remove_dealii_compatibility_fix’
   b. Create commit
      i. ‘git add FILE’
      ii. ‘git commit -m ‘A short message describing the change’
   c. Push and open PR
      i. ‘git push origin remove_dealii_compatibility_fix’
      ii. Open PR on github (CTRL-Click on shown link)
   d. Wait for review
   e. Address review (repeat steps b,c,d)
   f. Success!

Now repeat the steps in 3. on your own.
Report on projects the participants worked on

Reorganize the documentation to an automatic online build

Nick Featherstone, Maria Weber, Lorraine Hwang, Rene Gassmoeller

Rayleigh’s documentation was kept as a set of Latex files that could be compiled into .pdf documents. These pdf documents were stored inside the repository, which is non-ideal since they contain binary data. Also the diagnostic codes of Rayleigh are useful if they are available online and always up-to-date. Therefore we converted the Rayleigh documentation and now use the Sphinx documentation generator to automatically upload a html version of the documentation to [https://readthedocs.org/projects/rayleigh-documentation/](https://readthedocs.org/projects/rayleigh-documentation/).

Sphinx can also build .pdf versions of the documentation that are formatted like the existing manual. This way the documentation can be archived, published and referenced like a normal manual.

The ‘Rayleigh Diagnostic Values’ manual Sections 1-3 has been converted into a Sphinx html format (see middle figure above) as have the diagnostic code tables (see right figure above).
Add an automatic regression tester

Rene Gassmoeller

Rayleigh already had an automatic build tester that would check for compile-time errors for every pull request. We now added a regression test suite that runs a number of simple test cases every time a change is proposed and compares the current results with known 'good' results. Since the tests need to run fast, they usually do not run the benchmark to full equilibrium, but instead compare to an earlier state of a run. The reference model was also run to steady state and compared against published reference results.

Documentation Updates

Lorraine Hwang

Publications

The online documentation now has a publications list for publications using the code Rayleigh. Add your publications by altering the appropriate publicationsYYYY.bib file AND the publications.rst file. Publications are sorted by year.

NOTE:
- that each year has its own .bib file.
- References appear in the order they appear in the .bib file
- You must add the directive :cite:`bibkey` in the .rst file
Video Gallery

Updated formatting so that images tile 3 across.

Custom reference states and nondimensionalizations, with Jupyter notebook examples

Nicholas Nelson, Philipp Edelmann, Ryan Orvedahl, Lydia Korre, Loren Matilsky, Nick Featherstone, Mike Calkins

One outcome of the workshop was that Rayleigh's custom-equation coefficient interface was completed. This addition allows users to specify alternative nondimensionalizations. For example, they may nondimensionalize using the free-fall time instead of the viscous diffusion time. Additionally, the non-constant equation coefficients, which are primarily used in anelastic simulations, may now be freely specified. This allows users to specify custom reference states that differ from those provided in Rayleigh, and which may derive from alternative modeling software (e.g., stellar models evolved from the MESA code).

These customizations are enabled through both the main_input file and via a Python interface. For the latter, we developed a set of Jupyter notebooks that walk users through generating a custom-equation-coefficient file. These notebooks provide examples for specifying reference states already provided by Rayleigh (for testing purposes). We also developed notebooks that describe how to specify a nondimensional anelastic reference state and how to convert a stellar model, generated via the MESA code, into a Rayleigh equation coefficients file. Screen captures of the MESA and anelastic notebooks are shown in the two images below.
This script will take a MESA stellar evolution profile and convert it into a format that can be read in as a custom reference state in Rayleigh. You will need the rayleigh_diagnostics.py, reference_tools.py, and mesa.py files. You will also need a suitable MESA profile file, such as mesa.profile.

```python
In [ ]:
import numpy as np
import matplotlib.pyplot as plt
import scipy.interpolate as spint
import scipy.signal as ssig
import reference_tools as rt
import rpy2
import rayleigh_diagnostics as rd
import mesa
%matplotlib inline
```

```python
In [ ]:
def interp(r, v):
    prad = 10**p.log10(r**11) * mesa.rsol
    return np.interp(r, prad, v**11)
```

Set the `work_dir` variable to the location of the Python files listed above and MESA profile you wouldlike to use.

```python
In [ ]:
work_dir = '/Users/simelson/Research/Rayleigh/Rayleigh/tmp'
sys.path.append(work_dir)
```

```python
In [ ]:
p = mesa.profile('profile17.data')
```

Choose a suitable number of radial grid points. They do not need to be regularly spaced. You should err on the side of high resolution since Rayleigh's Chebyshev domains have very fine grid spacing at the top and bottom of the domain.

```python
In [ ]:
nr = 5000
r0 = 4.55e10 # in cm
r1 = 6.00e10 # in cm
radius = np.linspace(r0, r1, nr)
```

From the MESA model, Rayleigh will need the density, buoyancy function \( \rho g \), temperature, viscosity, thermal diffusion, electrical resistivity (for magnetic cases), heating profile (for cases with \( Q \neq 0 \)), entropy gradient (for cases with reference state advection). Note that MESA radial indices start at the bottom, while Rayleigh radial indices start at the top.
Jupyter Notebooks for visualization with YT

Maria Weber, Nick Featherstone

YT is a community-developed analysis and visualization toolkit for volumetric data utilizing Python. We have developed two notebooks for volume rendering with yt, which load in Rayleigh Spherical_3D output quantities interpolated onto a Cartesian grid. These can be found in /Rayleigh/post_processing, and are called YT_volrender_tutorial1.ipynb and YT_volrender_tutorial2.ipynb. There are still some known issues with YT, including difficulty rendering negative quantities. We will update these notebooks in the future once these issues have been resolved by the yt developers.

YT_volrender_tutorial1.ipynb shows how to render a volume with a custom transfer function that behaves like an isosurface. Shown is the radial velocity field from a Rayleigh benchmark. Only positive radial velocity values are rendered. Also shown is the custom transfer function used for the image. An example on how to rotate the
camera is also included, in addition to a tutorial on how to specify a transfer function with a user-defined opacity function. Note that these are examples only, and do not reflect the best transfer function choices for publications, etc.

YT_volrender_tutorial2.ipynb shows how to convert Spherical volumetric quantities into Cartesian quantities. Shown is the computed z velocity field from a Rayleigh simulation, looking down from the North pole. Only positive values are rendered. Also shown is the custom transfer function used for the image, similar in style to an isosurface. An example on how to rotate the camera to the pole is included.

Parallel routine to interpolate spherically gridded volumetric Rayleigh output onto a Cartesian grid

Kyle Augustson, Maria Weber, Nick Featherstone

Included in /Rayleigh/post_processing/kyle_interp is a parallel code to take Rayleigh Spherical_3D quantities and interpolate them onto a Cartesian grid. This is useful for visualization purposes. Makefiles and an input file are also included. Instructions on how to run and compile this code can be found in the Jupyter notebooks in /Rayleigh/post_processing called YT_volrender_tutorial1.ipynb and YT_volrender_tutorial2.ipynb.

Statistics about Rayleigh’s growth during the hackathon

The following contains a number of statistics about how much Rayleigh has grown during the hackathon (between July 22nd 2019, commit 84b2d38 and Aug 2nd 2019, commit fd907b2 to allow for late merges):

- Number of source files in Rayleigh before/after: 231 -> 241 +10
- Lines of code in Rayleigh before/after: 98440 -> 97341 -1,099
Comment: This is a good thing. Removing source code lines while adding functionality means we did a lot of cleanup of unnecessary code. In terms of all lines changed (not just source code) these are the changes:
11894 insertions(lines added), 48087 deletions (lines deleted)
Computed using: git diff 84b2d38 fd907b2 --stat
- Number of merged pull requests before/after: 85 -> 175 +90
- Commits in github before/after: 220 -> 443 +223
- Number of tests before/after: 2 -> 4 +2

These statistics were generated through the following commands:
- find . | egrep \.(F|F90|c|py|ipynb)$ | wc -l
- cat `find . | egrep \.(F|F90|c|py|ipynb)$` | wc -l
- git log --format=oneline | grep "Merge" | wc -l
- git log --format=oneline | grep -v "Merge" | wc -l
- ls -l tests/* | wc -l