

Rayleigh Hackathon 2024 Report

Logistics	1
Introduction	2
Land Acknowledgement	3
Timeline	3
Participants and areas of interest	5
Resources	7
Git Tutorial:	7
Visual Studio Code Tutorial	8
Report on projects the participants worked on	9
Fixed the conda development environment for MacOS	9
Rene Gassmoeller, Philipp Edelmann, Tami Rogers, Nick Featherstone	9
Fixed the automatic container build for x86 and Mac ARM architectures	10
Rene Gassmoeller	10
Containers for the NSF/TACC supercomputers	10
Rene Gassmoeller	10
Added an image gallery and updated the Rayleigh documentation	11
Brandon Lazard	11
Added a Pseudo-Incompressibility option	11
Brad Hindman, Loren Matilsky, Nick Featherstone	11
Updated Documentation to Include Streamfunction Equations solved by Rayleigh	12
Tami Rogers & Philipp Edelmann	12
Included “Newtonian Cooling” (Heating function)	12
Nick Featherstone & Tami Rogers	12
Modified the Configuration Script to Check for the Correct Combination of Libraries Specified	13
Brandon Lazard, Rene Gassmoeller, Nick Featherstone	13
Added a Function to read in the Rayleigh Checkpoint Files	13
Rathish Ratnasingam, Nick Featherstone	13
Added a Jupyter Notebook to read in the Rayleigh Checkpoint Files and Plot its output	13
Brandon Lazard, Nick Featherstone	13
Parallel I/O Bug Fixes	13
Nick Featherstone	13
Extra Scalar Fields	14
Cian Wilson	14
Coupled Boundary Conditions	14
Cian Wilson	14

Custom Reference States for Extra Scalar Fields	14
Cian Wilson, Nick Featherstone	14
Update publication lists	14
Lorraine Hwang	14
Allowed Rayleigh to increase resolution in multiple Chebyshev domains	15
Loren Matilsky + Nick Featherstone	15
Fixed small logical issue with the constants in the “equations Rayleigh solves” part of documentation	15
Loren Matilsky + Nick Featherstone	15
CMake build system alternative	15
Philipp Edelmann + Rene Gassmoeller	15
GPU prototyping	15
Philipp Edelmann + Tami Rogers	15
2024 Statistics about Rayleigh’s growth during the hackathon	16

Logistics

Slack:

https://join.slack.com/t/rayleighworkspace/shared_invite/zt-1fy7the9g-l80zezN~0bf64mFp2vKRiW

Documentation Outline

<https://docs.google.com/document/d/1qxnBEpvC8UMeh98dMDcbBtHp8yIC9YM4RXpJREB1Dms/edit?usp=sharing>

Introduction

For the 2024 Rayleigh hackathon, 10 user-developers of Rayleigh worked in an in-person hackathon over a 5-day period in Granby, CO. Attendees included a mix of new and veteran users, with career stages spanning from early to late career. Experts from geophysics, solar/stellar physics, and planetary atmospheres were all represented at this year's workshop. Several improvements to the source code, documentation, and supporting analysis routines were completed and/or initiated during this year's week-long workshop.

A major effort of this year's workshop was to complete the implementation *and documentation* of Rayleigh's "multiple-scalar-field" mode. When active, this mode of Rayleigh allows the advection-diffusion equation to be solved for additional scalar quantities (analogous to entropy or temperature). Those scalar variables act as passive tracers of the flow and, if desired, can be coupled via buoyancy to the momentum equation, making them active contributors to the dynamics. In addition, they can couple, in an arbitrary, user-defined fashion, to any of the other system variables at the boundaries. These changes allow, among other things, for Rayleigh to better represent the compositional convection and subsequent plating of iron onto the Earth's solid inner core. Discussions around this project began pre-2020, and the initial development began at the 2022 hackathon. This functionality is now fully-implemented for the Boussinesq mode of Rayleigh. As we are still thoroughly testing these new additions, documentation appears in the "Under Development" section of the User Guide for now.

Programming for another large addition to the physics supported by Rayleigh was nearly completed at this year's hackathon. The goal of that effort was to enable the use of the so-called "pseudo-incompressible" approximation in Rayleigh. This approximation would act as an alternative to the Boussinesq and anelastic approximations already admitted by the code. As with those two approximations, the pseudo-incompressible approximation similarly assumes a low-Mach number flow. At the same time, it places fewer restrictions on the size of thermal perturbations with respect to the background temperature or entropy – making it particularly suitable for the study of convective overshoot. An initial draft of the new code and associated documentation was completed at this year's workshop, and the new additions are now undergoing careful testing. We plan for these additions to be incorporated into the main codebase by this fall's release.

Worth highlighting also is another project which was a bit more diffuse in that it was composed of several smaller subprojects and pull-requests, involved improvements to the robustness of Rayleigh's build system and containerization. Rayleigh's cross-system support was improved by transitioning the conda development environment's math dependencies from MKL to OpenBLAS/FFTW3. This change enables seamless support on both PC and Mac systems. In addition, updated containers were produced for both the CI system employed by Rayleigh and the large-scale systems at the Texas Advanced Computing Center (TACC). Finally, as our user

base has grown, so too has the list of operating systems and compilers available to our users. While the configure script works well for most Intel and AMD systems, CMake support was also added at this year's workshop to facilitate porting Rayleigh to system configurations not envisioned by the configure script.

In addition to these larger projects, a number of smaller projects were undertaken that substantially improved the code's usability. These included updates to the documentation, updates to the notebook examples, several bug fixes, and continued exploration of the potential for using GPUs. Those efforts are documented in more detail below. All new additions to the codebase resulting from this workshop will be wrapped into the Rayleigh 1.3 release later this fall. 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.

Land Acknowledgement

We acknowledge the indigenous people and land in which we are gathered. Granby, Colorado has been home to the Nuu-agma-tuvv-pu (Ute) and Tsésthó'e (Cheyenne) peoples who have provided stewardship of this land over many centuries. We are honored and grateful to be here today on their traditional lands.

See: <https://landacknowledgements.org/native-lands/>

Timeline

Day	Scheduled items
Sunday 06/16	Arrival.
Monday, 06/17	9 am: Introductions. Morning rounds 10:30 am: Git + pull request tutorial, VS Code
Tuesday, 06/18	9:00 am: Morning rounds
Wednesday, 06/19	9:00 am: day off
Thursday, 06/20	9 am: Morning rounds
Friday, 06/21	9 am: Morning rounds
Saturday, 06/22	Departure by 10A

Participants and areas of interest

Name, affiliation, email	Goals and interests for this hackathon
Rene Gassmoeller, University of Florida rene.gassmoeller@mailbox.org	<ol style="list-style-type: none"> 1. Help others with their goals 2. Review pull requests 3. Create singularity container for TACC systems 4. Create an online hubzero tool? 5. Maybe: run some models
Lorraine Hwang UC Davis ljhwang@ucdavis.edu	<ol style="list-style-type: none"> 1. Logistics 2. Baking 3. Update citations
Nick Featherstone Southwest Research Institute nicholas.featherstone@colorado.edu	<ol style="list-style-type: none"> 1. Introducing people to Rayleigh's design 2. Helping others
Cian Wilson Carnegie Science cwilson@carnegiescience.edu	<ol style="list-style-type: none"> 1. Document scalar fields 2. Finalize coupled bc pull request 3. Scalar fields with custom reference states
Rathish Previn Ratnasingam Newcastle University, UK rathish.ratnasingam@ncl.ac.uk	<ol style="list-style-type: none"> 1. Introduce FD formulation for MHD, anelastic calculations
Philipp Edelmann Los Alamos National Laboratory pedelmann@lanl.gov	<ol style="list-style-type: none"> 1. Prototype some GPU code (OpenMP) 2. add CMake as another build system option
Bradley Hindman University of Colorado hindman@colorado.edu	<ol style="list-style-type: none"> 1. Develop a pseudo-incompressible formulation of the equation sets 2. Add additional outputs for the magnetic induction equation
Brandon Lazard University of California bjlazard@g.ucla.edu	<ol style="list-style-type: none"> 1. Add image gallery for Rayleigh 2. Learn, Learn, Learn
Tami Rogers	<ol style="list-style-type: none"> 1. Updating documentation 2. Working on getting similar code to Rayleigh on gpu 3. Looking into putting tracer particles into post processing

	4. Using Rayleigh for Hot Jupiters?
Loren Matilsky UC Santa Cruz loren.matilsky@gmail.com	<ol style="list-style-type: none">1. New terms (including non-LBR) in momentum equation + energy equation2. Allow changes in radial resolution (including multiple domains).3. Update documentation for equation sets.4. Add new time-scale choices for nondimensional anelastic reference_type = 5.5. Look at magnetic induction equation outputs with Brad/Nick.6. Try to implement r=0 coordinate singularity.

Resources

Git Tutorial:

- The slides from an earlier presentation:
<https://www.dropbox.com/s/6xvb4pyq7mefxp7/Git-Github-introduction.pdf?dl=0>
 - Git commands cheat sheet: <https://education.github.com/git-cheat-sheet-education.pdf>
 - Github workflow: <https://guides.github.com/introduction/flow/>
 - Git tutorial: <https://swcarpentry.github.io/git-novice/>
1. Explain and set up Git:
 - a. <https://swcarpentry.github.io/git-novice/01-basics.html>
 - b. <https://swcarpentry.github.io/git-novice/02-setup.html>
 2. Explain and setup Visual Studio Code:
 - a. <https://code.visualstudio.com/download>
 3. Explain Github Workflow:
 - a. <https://guides.github.com/introduction/flow/>
 - b. Ensure forked repositories
 - c. Ensure proper remotes
 4. Walkthrough
 - a. Create Branch
 - i. 'git checkout main
 - ii. 'git pull upstream main
 - iii. 'git checkout -b branch_name'
 - 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 branch_name
 - 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.

Visual Studio Code Tutorial

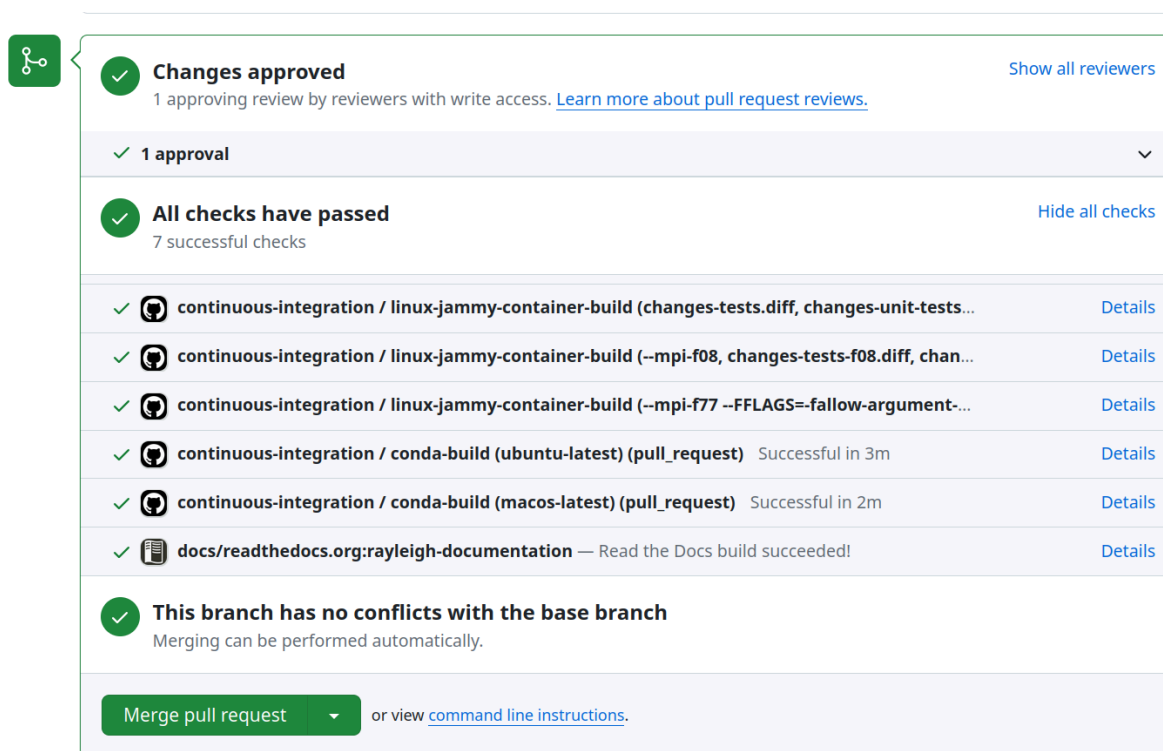
- VS Code is an Integrated Development Environment (IDE) that simplifies programming
- It is free, powerful, and used by the majority of open-source software developers
- If you are already comfortable with a different IDE stick to it, if you do not use an IDE so far, please install VS code
- It is simpler for us to explain and help you if most of us use the same IDE
- How to get: <https://code.visualstudio.com/>
- Documentation: <https://code.visualstudio.com/docs>
- Necessary/Useful extensions for this hackathon:
 - <https://marketplace.visualstudio.com/items?itemName=eamodio.gitlens>
 - <https://marketplace.visualstudio.com/items?itemName=MS-vsliveshare.vsliveshare>
 -

Report on projects the participants worked on

Fixed the conda development environment for MacOS

Rene Gassmoeller, Philipp Edelmann, Tami Rogers, Nick Featherstone

The Rayleigh conda environment contained the Intel MKL package, which is not supported for modern MacOS hardware architectures. In addition the configure script did not always select the correct standard C++ library on MacOS. All issues with the default Rayleigh installation instructions for Mac have been fixed and Intel MKL has been replaced with OpenBLAS as the default BLAS/LAPACK implementation.



The screenshot displays a GitHub pull request interface. At the top left is a green share icon. The main content area is a light blue box with a green border. It features several sections: 1. A green checkmark icon followed by the text 'Changes approved' and '1 approving review by reviewers with write access. [Learn more about pull request reviews.](#)' and a 'Show all reviewers' link. 2. A section for '1 approval' with a dropdown arrow. 3. A green checkmark icon followed by 'All checks have passed' and '7 successful checks' and a 'Hide all checks' link. 4. A list of CI jobs, each with a green checkmark, a GitHub Actions icon, a job name, and a 'Details' link. 5. A green checkmark icon followed by 'This branch has no conflicts with the base branch' and 'Merging can be performed automatically.' 6. A green button labeled 'Merge pull request' with a dropdown arrow, followed by 'or view [command line instructions.](#)'

Job Name	Status	Details
continuous-integration / linux-jammy-container-build (changes-tests.diff, changes-unit-tests...	Successful	Details
continuous-integration / linux-jammy-container-build (--mpi-f08, changes-tests-f08.diff, chan...	Successful	Details
continuous-integration / linux-jammy-container-build (--mpi-f77 --FFLAGS=-fallow-argument-...	Successful	Details
continuous-integration / conda-build (ubuntu-latest) (pull_request)	Successful in 3m	Details
continuous-integration / conda-build (macos-latest) (pull_request)	Successful in 2m	Details
docs/readthedocs.org:rayleigh-documentation — Read the Docs build succeeded!	Successful	Details

Fixed the automatic container build for x86 and Mac ARM architectures

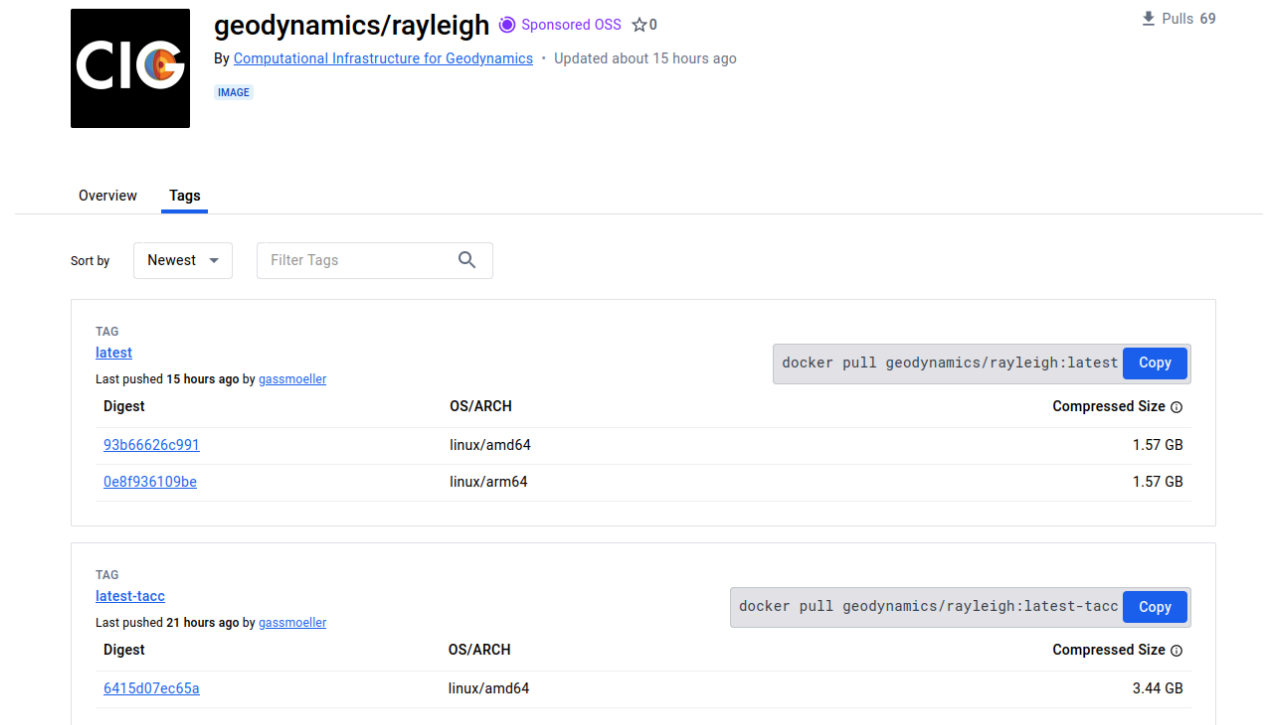
Rene Gassmoeller

I fixed the automatic build process for the Rayleigh docker images that are built and distributed for both x86 and ARM64 architectures. The images are now automatically uploaded and distributed via DockerHub and GitHub Container Registry.

Containers for the NSF/TACC supercomputers

Rene Gassmoeller

Rayleigh now builds and distributes docker containers that are compatible with the container system *Apptainer* on the NSF/TACC supercomputers Stampede3 and Frontera. These containers are available at <https://hub.docker.com/r/geodynamics/rayleigh/tags> and documented in the Rayleigh documentation (https://rayleigh-documentation.readthedocs.io/en/latest/doc/source/User_Guide/getting_started.html#using-the-apptainer-container-system).



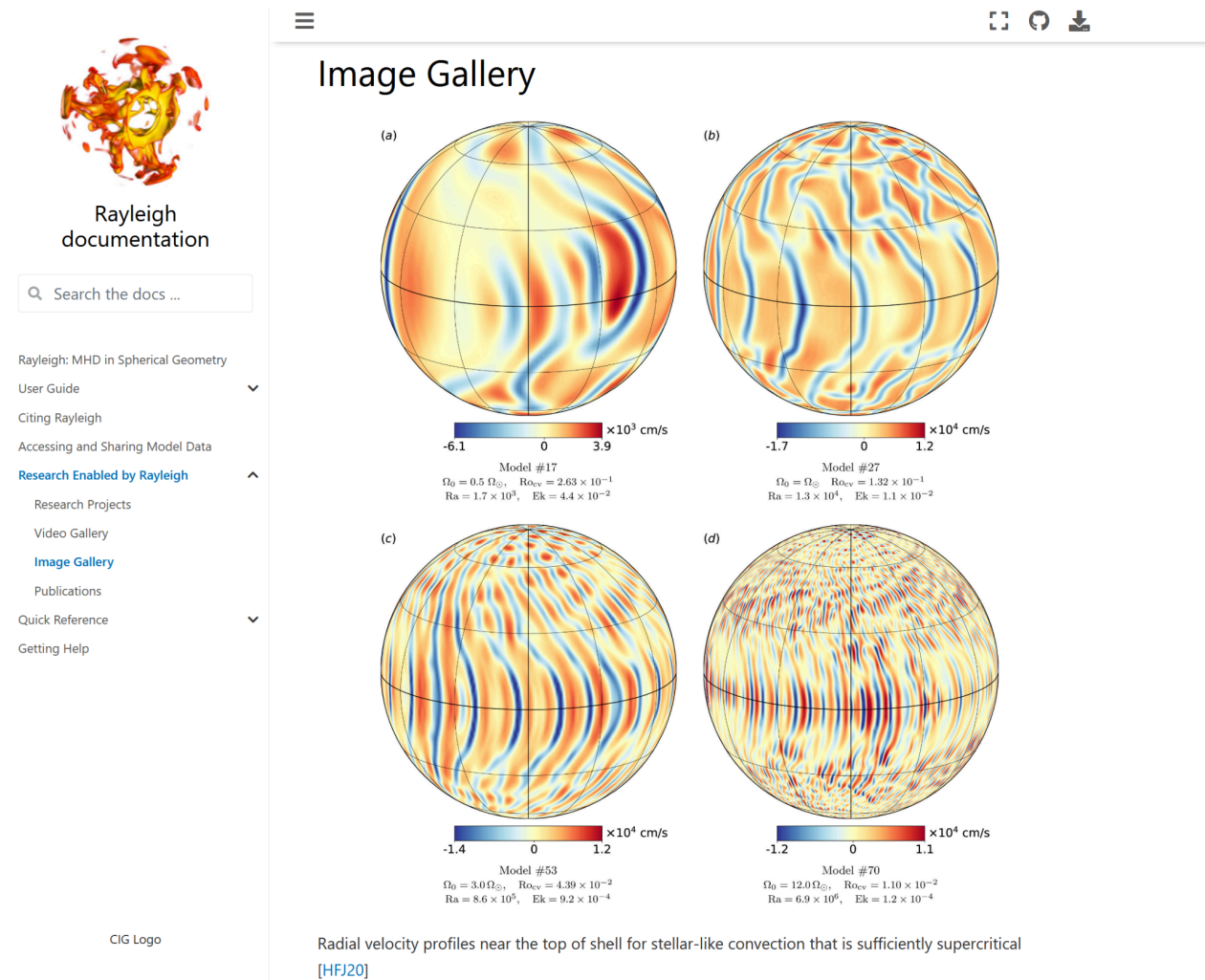
The screenshot shows the Docker Hub page for the repository `geodynamics/rayleigh`. The repository is sponsored by Computational Infrastructure for Geodynamics and was updated about 15 hours ago. It features two tags: `latest` and `latest-tacc`. The `latest` tag has two architectures: `linux/amd64` (1.57 GB) and `linux/arm64` (1.57 GB). The `latest-tacc` tag has one architecture: `linux/amd64` (3.44 GB). The interface includes a search bar, sort options (set to 'Newest'), and a 'Filter Tags' input field.

Tag	OS/ARCH	Compressed Size
latest		
Last pushed 15 hours ago by gassmoeller		
Digest	OS/ARCH	Compressed Size
93b66626c991	linux/amd64	1.57 GB
0e8f936109be	linux/arm64	1.57 GB
latest-tacc		
Last pushed 21 hours ago by gassmoeller		
Digest	OS/ARCH	Compressed Size
6415d07ec65a	linux/amd64	3.44 GB

Added an image gallery and updated the Rayleigh documentation

Brandon Lazard

An [image gallery](#) has been added to the documentation, and several images from published papers using Rayleigh have been added to the gallery (with permission from the authors). As part of this process, the [publications](#) page in the documentation was updated through 2023.



The screenshot shows the Rayleigh documentation website. On the left is a navigation sidebar with a search bar and a list of links including 'User Guide', 'Citing Rayleigh', 'Research Enabled by Rayleigh', and 'Image Gallery'. The main content area is titled 'Image Gallery' and displays four spherical plots (a, b, c, d) showing radial velocity profiles. Each plot includes a color scale and associated model parameters.

Model #17
 $\Omega_0 = 0.5 \Omega_{\odot}$, $Ro_{cv} = 2.63 \times 10^{-1}$
 $Ra = 1.7 \times 10^3$, $Ek = 4.4 \times 10^{-2}$

Model #27
 $\Omega_0 = \Omega_{\odot}$, $Ro_{cv} = 1.32 \times 10^{-1}$
 $Ra = 1.3 \times 10^4$, $Ek = 1.1 \times 10^{-2}$

Model #53
 $\Omega_0 = 3.0 \Omega_{\odot}$, $Ro_{cv} = 4.39 \times 10^{-2}$
 $Ra = 8.6 \times 10^6$, $Ek = 9.2 \times 10^{-4}$

Model #70
 $\Omega_0 = 12.0 \Omega_{\odot}$, $Ro_{cv} = 1.10 \times 10^{-2}$
 $Ra = 6.9 \times 10^6$, $Ek = 1.2 \times 10^{-4}$

Radial velocity profiles near the top of shell for stellar-like convection that is sufficiently supercritical [HFJ20]

Added a Pseudo-Incompressibility option

Brad Hindman, Loren Matilsky, Nick Featherstone

The anelastic approximation is only self-consistent in atmospheres that have a nearly adiabatic stratification. The pseudo-incompressible approximation, however, is one that works both in stable (i.e., nonadiabatic) regions and convection zones. The approximation is primarily a modification of the continuity equation, which means that the stream function implementation used by Rayleigh needed to be modified.

We made all of the required changes to implement the pseudo-incompressible approximation in Rayleigh. The new option appears as an alternative way to run the anelastic system (since the two approximations are similar). Making these changes involved adding several new reference state variables that track the entropy and its radial derivatives. Further, throughout the Physics modules, additional terms and multiplicative factors needed to be added that acknowledge entropy variation in the background. Finally, we added a term in the buoyancy that is normally ignored under the LBR formulation of the anelastic approximation (it is proportional to the background entropy gradient). In the future, it should be possible to add a flag that would instruct the anelastic mode to also include this term, if desired. We are still in the process of testing this functionality, and making sure that errors in Rayleigh's native anelastic mode were not created unintentionally. Once this testing is complete, the pseudo-incompressible code will be wrapped into the main branch of Rayleigh (hopefully late summer/early fall of 2024).

Updated Documentation to Include Streamfunction Equations solved by Rayleigh

Tami Rogers & Philipp Edelmann

Previous documentation included the (magneto-)hydrodynamic equations cast in terms of velocity and magnetic field. The equations that are actually solved in Rayleigh are cast in terms of the stream/flux functions W, Z, P, A and C . While equivalent, their somewhat more complicated form has now been described in the documentation.

Included "Newtonian Cooling" (Heating function)

Nick Featherstone & Tami Rogers

An optional Newtonian cooling term and associated documentation was developed for Rayleigh this week. This appears to be working as planned and will be wrapped into the main codebase soon, once a published hot Jupiter model has been tested with Rayleigh.

Modified the Configuration Script to Check for the Correct Combination of Libraries Specified

Brandon Lazard, Rene Gassmoeller, Nick Featherstone

Previously, the user could specify an incorrect combination of libraries in their configuration command. This would result in the code approving the combination but during the compilation it would output an error at the end. To prevent this, we modified the configuration script to output an error if an incorrect combination of libraries is specified.

Added a Function to read in the Rayleigh Checkpoint Files

Rathish Ratnasingam, Nick Featherstone

Some of us spent the week writing a Python routine that can read in a Rayleigh checkpoint file. The checkpoint structure is similar to, but also somewhat different from the Shell_Spectra output structure. The latter stores the spectrum as a square array ($l_{\max}+1 \times l_{\max}+1$) and includes zero values for ell - m combinations with $m > ell$. The Checkpoint files are stored in a more compact format, omitting the zeros. This meant that the shell spectra function in `rayleigh_diagnostic.py` could not be used directly to read in a checkpoint. The new Python function is capable of reading a checkpoint file and returning a data structure consistent with that used for the Shell_Spectra class.

Added a Jupyter Notebook to read in the Rayleigh Checkpoint Files and Plot its output

Brandon Lazard, Nick Featherstone

Using the new `read_checkpoint` function in `rayleigh_diagnostics.py` (described above), we created a tutorial notebook that demonstrates how to read checkpoint files. The notebook walks the user through transforming the spectral checkpoint input into physical space and plotting the results.

Parallel I/O Bug Fixes

Nick Featherstone

We recently uncovered a bug in Rayleigh where the code attempted to update and close a file that failed to open. An initial fix of this bug introduced another (worse) bug that caused all parallel I/O to hang. This bug was fixed during the course of this workshop

Extra Scalar Fields

Cian Wilson

Extra scalar fields were added at a previous hackathon but not documented. This year we documented this functionality and added it to the “Under Development” section of the User Guide. This documentation includes a summary of the functionality that has been tested so far and a future development plan for extending this functionality to additional reference-state types.

Coupled Boundary Conditions

Cian Wilson

The final pieces of development for coupled boundary conditions were implemented during this hackathon. This included warning users of the change of previous default behaviors and the removal of unnecessary flags for passive (not coupled) fields. Documentation was built on top of the documentation for extra scalar fields in the “Under Development” section of the User Guide.

Custom Reference States for Extra Scalar Fields

Cian Wilson, Nick Featherstone

Extra scalar fields currently only work with a limited set of reference states (and have only been used with Boussinesq). Extending this to a wider range of applications requires that the user is able to specify custom reference states for the scalar fields, which is currently not implemented. In order to not interfere with the custom reference states for mandatory fields we devised a development plan for new custom constants (“d”) and custom functions (“g”) specifically for the extra scalar fields. This has been included in the documentation and development has begun but not completed.

Update publication lists

Lorraine Hwang

We aligned the geodynamics.org citations with Rayleigh publications.

Allowed Rayleigh to increase resolution in multiple Chebyshev domains

Loren Matilsky + Nick Featherstone

We modified `src/Physics/Checkpointing.F90` to allow the user to increase radial resolution even for multiple Chebyshev domains (e.g., a checkpoint created with `ncheby=64,64,64` could be restarted with `ncheby = 96,128,96`). Before, the code would effectively scramble the Chebyshev coefficients between different subdomains.

We also caused Rayleigh to throw an error and exit when the user tried to decrease radial resolution in any subdomain (the error and exit apply even to the single-domain case). Before, even in one domain, the code simply would not “do anything” if the user tried to lower radial resolution, with the result that all fields were initialized to zero and the code would happily integrate forward.

Fixed small logical issue with the constants in the “equations Rayleigh solves” part of documentation

Loren Matilsky + Nick Featherstone

We made it clear in the documentation that $c_9 = c_8 * c_4 * c_7$ (which multiplies the Ohmic heating term) is not an independent constant, if the equations are to be consistently nondimensionalized.

CMake build system alternative

Philipp Edelmann + Rene Gassmoeller

We can now use CMake as an alternative to the configure script. The main advantage of this is that we get support for new compilers and libraries for “free”. We also added CI tests for this method on Ubuntu and macOS.

GPU prototyping

Philipp Edelmann + Tami Rogers

We experimented with offloading part of the workload to GPUs in a simple anelastic code similar to Rayleigh. We tested with OpenMP target offloading and the `nvfortran` and `gfortran` compilers. In early tests we saw that we were dominated by the copy operations because only few loops were moved to the GPU. This is expected to improve once more parts of the code are moved to

the GPU. Gfortran is stricter with its interpretation of constructs like defaultmap(none), which nvfortran largely seems to ignore.

2024 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 Jun 15 2024, commit 28dd05d and Jun 25 2023, commit f32a7e2 to allow for late merges):

- Number of source files in Rayleigh before/after: 170 -> 172 + 2
- Lines of code in Rayleigh before/after: 71741 -> 72699 + 958
- Number of merged pull requests before/after: 451 -> 494 + 43
- Commits in github before/after: 1370 -> 1499 + 129
- Number of tests before/after: 6 -> 8 + 2
- Lines of documentation before/after: 4664 -> 4950 + 286

Statistics from 2023:

- Number of source files in Rayleigh before/after: 166 -> 171 +5
- Lines of code in Rayleigh before/after: 67716 -> 72398 +4682
- Number of merged pull requests before/after: 406 -> 424 +18
- Commits in github before/after: 1242 -> 1309 +67
- Number of tests before/after: 6 -> 6 +0

These statistics were generated through the following commands:

- To checkout a certain date:
`git checkout `git rev-list -n 1 --first-parent --before="2024-06-15 12:00" main``
- `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`
- Tests were manually counted
- `cat `find -L doc/source | egrep '\.(rst)$'` | wc -l`



Left to right: Philipp Edelman, Tami Rogers, Brad Hindman, Nick Featherstone, Rene Gassmoeller, Loren Matilsky, Rathish Previn Ratnasingam, Cian Wilson, Brandon Lazard