Rayleigh Hackathon 2023 Report

Logistics

The hackathon occurs in person in Golden, Colorado. Participants that have to self-isolate due to Covid will use the following Zoom Link:

https://ufl.zoom.us/j/94646279339?pwd=SzhoUlFwSUNxWmZOTFRZMGtnTGVIZz09

Slack:
https://join.slack.com/t/rayleighworkspace/shared_invite/zt-1fy7the9g-I80zezN~0bf64mFp2vKRiw

Logistics spreadsheet:

Dataset Listing:
https://osf.io/j275z/wiki/Dataset%20Listing/

Link to last year's report:
https://geodynamics.org/resources/1929/download/Rayleigh_Hackathon_2021_Log.pdf

A Miro Board for Documentation Flowcharting/Brainstorming:
https://app.conceptboard.com/board/ksgb-gmck-7011-phmc-2ysh

Notes on Documentation Discussion
https://docs.google.com/document/d/17TthEITd0y4WZw0uriRj96YpolaKP6u-Cu6xi30y_gQ/edit?usp=sharing

Documentation Outline
https://docs.google.com/document/d/1qxnbEpvC8UMeh98dMDcbBtHp8ylC9YM4RXpJREB1Dms/edit?usp=sharing

Introduction

For the 2023 Rayleigh hackathon, 13 user-developers of Rayleigh worked in an in-person hackathon over a 5-day period in Golden, CO on the campus of the Colorado School of Mines. 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.

From the standpoint of the source code, two major efforts were undertaken and largely completed this year. The first of these involved the re-implementation of a radial finite-difference scheme in Rayleigh. This functionality has existed for some time, but was disabled in favor of the more accurate Chebyshev approach used for the radial discretization. The gridpoint locations used in the Chebyshev scheme are, however, overly constraining for some problems that have interior boundaries requiring fine resolution locally. This can be overcome through the use of a finite-difference scheme appropriate for non-uniform grids. Such a scheme, with 4th-order spatial accuracy, is now fully implemented in Rayleigh and has been brought up-to-date with the rest of the code base.

The second major effort was the development and testing of coupled boundary conditions for the multi-scalar field mode that was implemented in the 2022 hackathon. This new feature will allow, among other things, for Rayleigh to better represent the compositional convection and subsequent plating of iron onto the solid inner core. Both the 4th-order finite-difference scheme and the coupled boundary conditions are now fully implemented in Rayleigh. Appropriate documentation is now being finalized in preparation for the fall 2023 release.

Additionally, two projects of particular note were begun with an eye toward future applications of the Rayleigh software. The first of these was to begin implementing a new mode in Rayleigh that enables the use of the so-called pseudo-incompressible approximation as an alternative to the Boussinesq and anelastic approximations already available in Rayleigh. The pseudo-incompressible approximation similarly assumes low-Mach number flows, but places fewer restrictions on the assumed size of thermal perturbations about the background state – making it particularly suitable for the study of convective overshoot. The second new project involves the implementation of dynamic (in time) boundary conditions. This effort was motivated by the desire to incorporate a topographical perturbation to the otherwise spherical boundary. The framework developed for this effort also has broader potential, such as the coupling of a time-varying mantle-convection simulation to Rayleigh's upper boundary. Work on these two projects will continue throughout the coming year and will hopefully be finalized at next year's Hackathon.

In addition to source-code modifications, a number of modifications to the broader Rayleigh ecosystem were also undertaken, including updates to Rayleigh's containerization, the addition of new tutorial notebooks and a new benchmark test for the multiple-scalar-field mode now in Rayleigh. All new additions to the codebase resulting from this workshop will be wrapped into the Rayleigh 1.2 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. Golden, Colorado has been home to the Núu-agha-tʉvʉ-pʉ̲ (Ute) and Tséstho'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.

Timeline

Day	Scheduled items
Sunday 06/11	Arrival, Introductions
Monday, 06/12	9 am: Morning rounds 10:30 am: Git + pull request tutorial
Tuesday, 06/13	9 am: VS code tutorial 9:30 am: Morning rounds 2:30 pm: How to publish (Lorraine)
Wednesday, 06/14	9:30 am: Morning rounds 10:00 am: SPH tool presentation (Catherine)
Thursday, 06/15	9 am: Morning rounds
Friday, 06/16	9 am: Morning rounds
Saturday, 05/17	Departure

Participants and areas of interest

Name, affiliation, email	Goals and interests for this hackathon
Rene Gassmoeller, University of Florida rene.gassmoeller@mailbox.org	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	1. Logistics
Nick Featherstone Southwest Research Institute nicholas.featherstone@colorado.edu	1. Introducing people to Rayleigh's design 2. Helping others
Cian Wilson	1. Learn how to approach coupled bcs

Carnegie Science cwilson@carnegiescience.edu	2. Scalar fields with custom reference states 3. Standardize I/O for generic input
Rathish Previn Ratnasingam Newcastle University, UK rathish.ratnasingam@ncl.ac.uk	1. Introduce FD formulation for MHD, anelastic calculations
Philipp Edelmann Los Alamos National Laboratory pedelmann@lanl.gov	1. GPU port based on Ryan's work 2. integrate finite differences with Rathish 3. get Rayleigh into Spack mainline
Bradley Hindman University of Colorado hindman@colorado.edu	1. Pseudo-incompressible formulation for the continuity equation
Catherine Blume University of Colorado Boulder catherine.blume@colorado.edu	1. Documentation and examples using spectral_utils.py (spherical harmonic transforms, Chebyshev transformers, d/dr, d/dtheta, d/dphi in spectral and physical space)
Kyle Augustson Northwestern University kyle.augustson@northwestern.edu	1. Begin implementation of sparse matrix methods for the implicit portion of the solver. 2. Take a look at alternative time evolution methods (IMEX).
Peter Driscoll Carnegie Institution for Science pdriscoll@carnegiescience.edu	1. Derive equations that describe coupled boundary conditions for multiple scalar fields. 2. Run Rayleigh with a custom reference state for dimensionless anelastic.
Loren Matilsky UC Santa Cruz loren.matilsky@gmail.com	1. Discuss + modify custom reference framework (reference_type = 4) 2. Finish non-dimensional anelastic implementation (reference_type = 5) 3. Possibly: explore even-Chebyshev expansion to include r=0 coordinate singularity.
Tobias Oliver CU Boulder tobias.oliver@colorado.edu	1. Implement Topography boundary conditions into Rayleigh (testing)
Chi Yan cyan@carnegiescience.edu	1. Provide input namelists for the thermal-chemical dual convection benchmark to Breuer_2010_GJI. 2. Implement a stable layer in Boussinesq approximation using the custom reference (4).

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.

Report on projects the participants worked on

Fix Rayleigh's automatic docker image generation

Rene Gassmoeller

Rayleigh's docker image was supposed to be automatically rebuilt after every change to the master branch, but this workflow was broken by the name change of the main branch at the last hackathon. This is fixed now.

Custom reference framework discussion

Loren

1. To recap, there are four reference types that can specify the various c's and f's in the PDEs that Rayleigh solves. Reference_type = 1,2,3 have everything specified through keywords in main_input (these need no modification). Reference_type = 4 (custom reference) has by default everything specified through an input binary file.
2. Currently people are allowed some mixing and matching between reference_type = 4 and (e.g.,) nu_type = 1, nu_top = 1.0d12. This has caused confusion on
    a. What happens when conflicting specifications are made (e.g., nu_top = 1.0d012, ra_constants(3) = 5.3d12, something totally difference in binary file)
    b. What we *want* to happen for conflicting specifications.
3. We agreed on a fairly stringent modification that would basically force (for reference_type = 4) the user to specify everything from the input binary file.
    a. The only place the user will be able to override the binary file is through the "override_constants" framework (i.e., only constants, not functions, can be modified by the user after the fact).
    b. The user will *not* (no longer) be able to override the binary file using special keywords (e.g., nu_type, nu_top, Ekman_Number, Luminosity) associated with other reference types. This means Loren and Catherine's main_input files will break slightly :) But hopefully nobody else's!
    c. We need to decide on what to do with heating_type—I think we said it would be set to 10 (say) so it is always used by the code. If the user wants no heating, they need to explicitly set c_10*f_6 to zero via the binary file.
4. The "augment_reference" stuff (for reference_types = 1, 2, 3) we should also discuss at some point. I believe only gravity, heating, and f_14 (e.g., dS/dr, stable stratification) may be modified currently. Maybe we want to make it so anything can be modified, or at least the diffusivities can be?
5. To do:
    a. Modify code to adhere with a and b hierarchy (plus what we decide for c).
    b. Update documentation to reflect how this all works.

c. Make sure the code EXITs and throws an error if user specifies reference_type = 4 that is unspecified. Possibly throw warning that “user can’t use nu_type, nu_top with reference_type = 4 anymore” or something.
d. Make sure notebooks are up-to-date and show each use case.

Pseudo-Incompressible Approximation

Brad

1. Derived the evolution equations for the stream functions for the pseudo-incompressible equation set. (pressure and buoyancy finished, advection and viscosity still being worked out).

Sparse Methods and Time-stepping

Kyle Augustson

Decided on a representational basis and test basis for the operators and non-constant coefficients for the radial ODEs solved for each ell and m pair in the implicit portion of the code. Examined the code to see how to go about using Kronecker products to build the problem matrices within the constraints of the current code base, trying to minimize overall structural changes. Also I wrote a Mathematica workbook to compute the expansion of the stress tensor for the pseudo-incompressible equations as expressed on the new stream functions that use the potential density. I have written a jupyter notebook that uses the sparse Ultraspherical polynomial representation to construct test problems before beginning their implementation into Rayleigh to construct the implicit matrices. The workbook and notebook can be found here: https://github.com/kyle-augustson/Playing-with-Polynomials (it’s private for now, but let me know if you want access). The linear diffusion problem can also handle non-constant coefficients, which necessitated the implementation of a basis product transform akin to that of the Clebsch-Gordon coefficients but for Ultraspherical polynomials. I have written a separate notebook that deals with nonlinear problems (the Newell–Whitehead-Segel reaction-diffusion equation) to test the scheme. Also available on github. It works using the MCNAB2 scheme (see Ascher 1995 for details).

GPU port

Philipp Edelmann

Submitted a PR to get Rayleigh into Spack's main repo.
Added support for the Podman container runtime in this PR.

Got Ryan's work to build and run on Nvidia V100 GPUs.

Roadmap for Nvidia GPUs:

Rayleigh cannot be compiled with nvfortran because it lacks support for quad-precision numbers. This is used in Math_Layer/Legendre_Polynomials.F90. Just compiling this file with gfortran is not an option due to the incompatible .mod file formats.

gfortran has support for OpenACC and can generate code to offload to Nvidia and AMD GPUs. I have tested this successfully, but it would involve building our own gfortran on every cluster we use with Nvidia GPUs.

Opened an issue with all the findings of this week to discuss the roadmap.

Finite-Difference method and MHD in Rayleigh

Rathish Previn Ratnasingam

Updated Rayleigh's FD formulation and its compatibility with anelastic, MHD benchmarks. We started out with figuring out why the Jones MHD benchmark was failing. Two problems were found; the uniform radial grid was set up incorrectly and the L conservation at the boundary only applied for chebyshev grids. The fixes made were first, use a uniform dr grid when no dr is set up in the main_input file and second, set the weights for L integration to FD weights. All the benchmarks are now being tested.

Update Rayleigh docker image to work with Singularity/Apptainer and Apple Silicon

Rene Gassmoeller

Rayleigh's docker image was updated to be compatible with the Singularity/Apptainer container system that is used on many HPC computing resources. To use cluster specific networks the image still needs to be adapted, but the base image should work on any cluster as long as only a single node is used. This change also includes removing outdated docker images for no longer supported operating systems. This change also includes building the docker image for the ARM64 CPU architecture, which greatly speeds up the execution of the container for Apple M1 and M2 processors.

Implement Topography Boundary Conditions to Rayleigh

Tobias Oliver

Want to incorporate capability to handle bc's that deviate from a spherical surface. The proposed method is to approximate a bc of (say) T(r = 1 + h(\theta,\phi)) = 0 as T(r=1) + h dT/dr|_1 = 0. This requires a non-zero RHS for the b.c. at T(r=1)
Method has been implemented but so far seems unstable, and is in testing. Note that this method couples the modes and thus has to be treated with the non-linear terms.

The routines implemented can be extended to dynamic boundary conditions. Dynamic bc's are probably easier from a stability standpoint and could be included in the implicit solve rather than treating as a non-linear term, but the current implementation will evaluate RHS of bc at previous timestep.

Dual Buoyancy System Benchmark against Breuer et al. (2010)

Chi Yan

1. In a system of separate thermal (T) and compositional (c) buoyancy sources, we need a definition of Rayleigh # that is independent of thermal or compositional diffusivities (κ_T, κ_C). Therefore, we use the following Ra # definition that is written as:

$Ra^T_{vis} = \frac{g_0 \alpha \Delta T d^3}{\nu^2}$ for thermal Ra # &, $Ra^C_{vis} = \frac{g_0 \alpha \Delta C d^3}{\nu^2}$ for compositional Ra #

Which can be related to conventional Ra # as:

$Ra^T = Pr^T \bullet Ra^T_{vis}$ & $Ra^C = Pr^C \bullet Ra^C_{vis}$

Where Pr^T & Pr^C are the thermal and compositional Prandtl numbers.

Through varying thermal/chemical buoyancy contributions in varying test cases, the total Ra # is kept the same. Let δ be the thermal contribution ($\delta \in [0, 1]$), then:

$$Ra^{tot}_{vis} = \delta Ra^T_{vis} + (1 - \delta) Ra^C_{vis} = \delta Pr^{T^{-1}} Ra^T + (1 - \delta) Pr^{C^{-1}} Ra^C$$

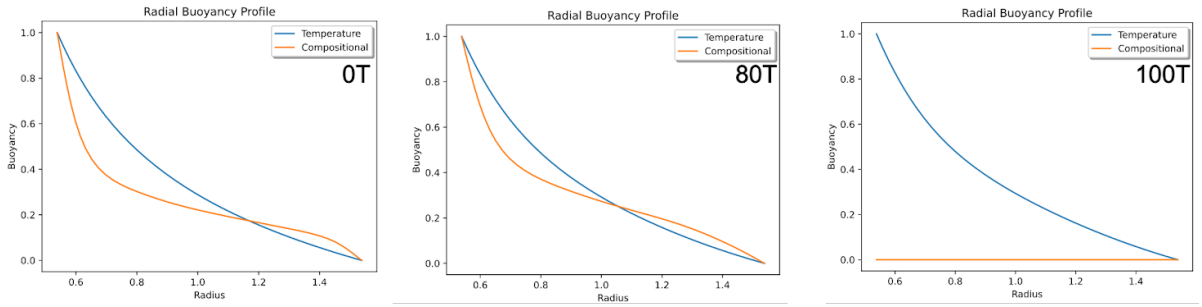
Knowing Ra^{tot}_{vis} and δ, one can get (what we need for input Ra parameters)

$Ra^T = \delta Pr^T Ra^{tot}_{vis}$ & $Ra^C = (1 - \delta) Pr^C Ra^{tot}_{vis}$

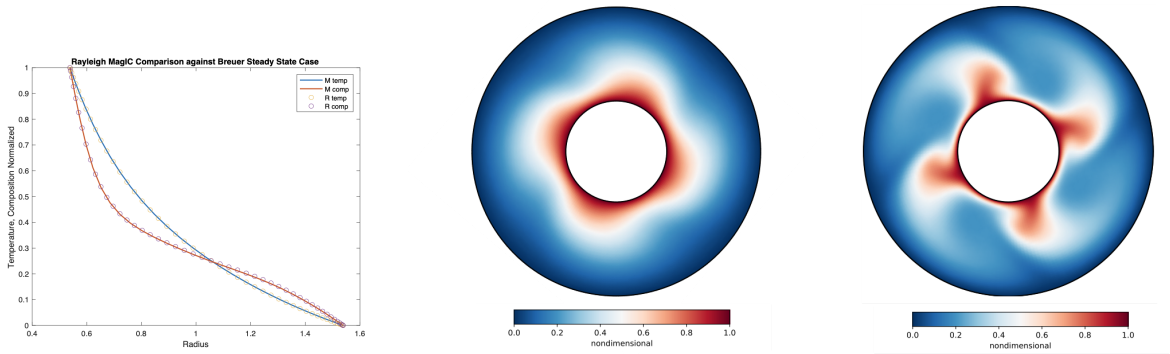
Case 0 Benchmark in Breuer_2010_GJI: $Ra^{tot}_{vis} = 2 \times 10^5$

δ	0	80	100
Ra^T	0	4.8E4	0
Ra^C	6E5	1.2E5	6E4

Please refer to **Table 1 in** Breuer_et_al (2010) to compare the diagnostic parameters.



Comparison of Rayleigh (lines) of the 80T-20C case (middle) with MagIC (circles)



Snapshots of the equatorial slices of the thermal (left) and compositional (right) profile.

---

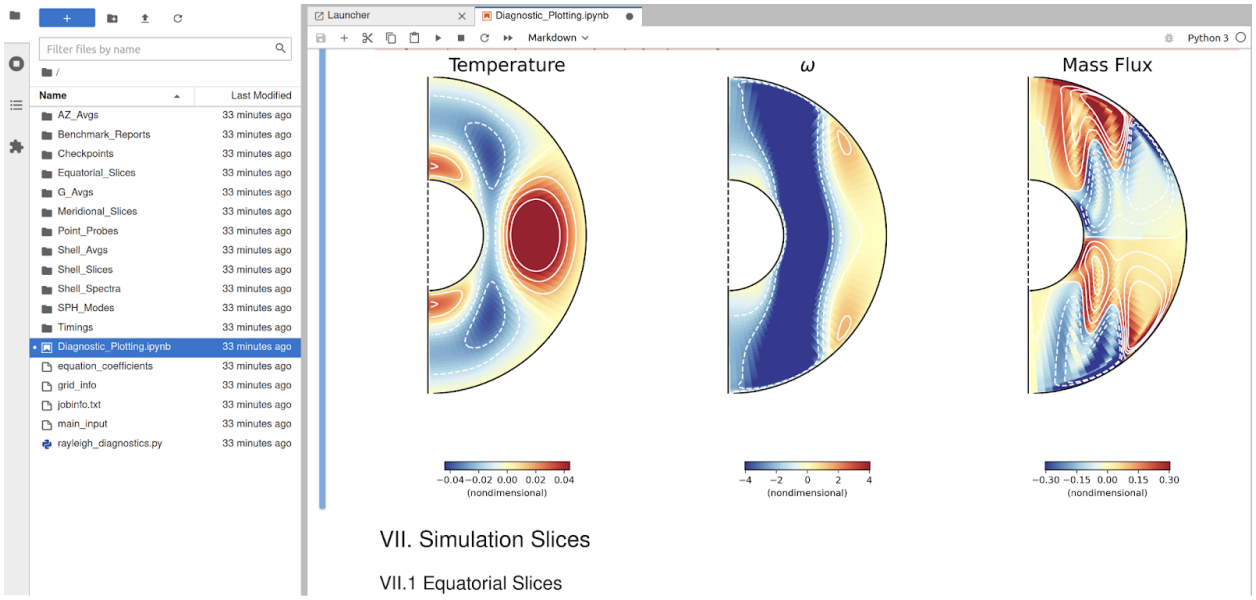
2. stably stratified layer (SSL) implementation in the Boussinesq system

The SSL is built through the background temperature gradient using custom_reference (4) profile. The description of the stable layer is done through the custom function f14 (refer to documentation), other functions and constants are implemented accordingly through Boussinesq approximation. Two types of stable layer are provided, one using step function, and the other using two tanh functions. A preliminary set of tests have been performed, such as minimal radial velocities in the stable layer region should be expected.

Online tool on geodynamics.org

Rene Gassmoeller

Added an online executable Jupyter notebook on the CIG website (geodynamics.org) that includes output from a small example model computed by Rayleigh. The notebook teaches how to analyze and visualize Rayleigh output.



Snapshot of the executable tool "Rayleigh Jupyter Notebooks" on geodynamics.org.

Added Jupyter notebook tutorial for spectral_utils.py

Catherine Blume

Spectral_utils.py in the post-processing directory provides tools to perform spherical harmonic, Legendre, Fourier, and Chebyshev transforms, along with radial and angular derivatives in both physical and spectral space. A tutorial jupyter notebook spectral_utils_tutorial.ipynb has now been added that provides examples using this tool. Because this makes converting between Shell Spectra and Shell Slices easy, the user will only need to output Shell Spectra, saving disk space. This may also reduce the necessity of outputting derivatives.

Contents

A. Case 1: Convert Shell Spectra to Shell Slices, $(l, m) \rightarrow (\theta, \phi)$

Instantiate class

```
# instantiate SHT class
SHT = spectral_utils.SHT(ntheta, spectral=False, dealias=dealias)

# alternate instantiation
# SHT = spectral_utils.SHT(lmax, spectral=True, dealias=dealias)
```

Perform transformation

```
# convert from spectral space to physical space
sample_spectra_to_slice = SHT.to_physical(sample_spectrum, th_l_axis=0, phi_m_axis=1)
```

Added new general non-dimensional anelastic reference_type = 5

Loren Matilsky

New reference_type = 5 allows user to specify a generalized non-dimensional anelastic reference state and non-dimensional control parameters.

This differs from reference_type = 3 by allowing for stable and unstable polytropes (user can now choose specific_heat_ratio and polytropic index poly_n separately), as well as several choices for non-dimensionalization:

1. User can choose to non-dimensionalize the polytrope at the inner boundary, outer boundary, or by volume-averages.
2. User can input arbitrary length-scale for non-dimensionalization (default is shell-depth)
3. Currently only the viscous diffusion time is used for the time scale, but the framework is set up so that other choices (e.g., 1/(2Omega_0)) can be easily implemented in the future simply by rescaling the appropriate constants by ratios of physical time-scales.

2023 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 11 2023, commit 1b99187 and Jun 29 2023, commit d8af7a0 to allow for late merges):

- 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

Statistics from 2022:

- Number of source files in Rayleigh before/after: 160 -> 163 +2
- Lines of code in Rayleigh before/after: 65195 -> 68465 +3270
- Number of merged pull requests before/after: 342 -> 377 +35
- Commits in github before/after: 1030 -> 1167 +137
- Number of tests before/after: 6 -> 6 +0

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
- Tests were manually counted


THAILAND
ASPECT