Code Review for Scientific Software

experiences building an online tutorial

Helen Kershaw
DAReS, NSF NCAR
Terminology

**BSSwF**  Better Scientific Software Fellowship

**NSF**  National Science Foundation
**NCAR**  National Center for Atmospheric Research
**UCAR**  University Corporation for Atmospheric Research
**SEA**  Software Engineering Assembly (UCAR/NCAR)
**DART**  Data assimilation Research Testbed
**DARes**  Data Assimilation Research Section
**AMS**  American Meteorological Society
Goals

- Tell you about my BSSwF project
- Share my experience building the tutorial
- Share practice and experience with code review from UCAR SEA
Goals

● Tell you about my BSSwF project
● Share my experience building the tutorial
● Share practice and experience with code review from UCAR SEA

● Get you to think about code review
Goals

- Tell you about my BSSwF project
- Share my experience building the tutorial
- Share practice and experience with code review from UCAR SEA

- Get you to think about code review
- And the joy of open source software
About me
About me

The Core
2003 · 2h 15min · PG-13
Action · Adventure · Sci-Fi
⭐ 5.5/10 ⭐ Rate

The only way to save Earth from catastrophe is to drill down to the core and set it spinning again.

Jonathan Mound's credits: Additional Crew (as Dr. Jonathan Mound)
About me
A Physics-Informed Neural Network for Quantifying the Microstructural Properties of Polycrystalline Nickel Using Ultrasound Data

A promising approach for solving inverse problems

We apply physics-informed neural networks (PINNs) to quantify the microstructure of polycrystalline nickel by computing the spatial variation of compliance coefficients (compatibility, stiffness, and rigidity) of the material. The PINNs are trained with multi-scale ultrasonic acoustic wavefield data acquired at an ultrasound frequency of 3 MHz for the polycrystalline nickel. The ultrasonic wavefield data are represented as a deformation on the top surface of the material with the deformation measured using the method of laser vibrometry. The ultrasonic data are further augmented with wavefield data generated using a finite-element-based solver. The neural network is physically informed by the in-plane and out-of-plane elastic wave equations, and the convergence is accelerated using adaptive activation functions. The overarching goal of this work is to infer the spatial variation of compliance coefficients of materials using PINNs, which can be used to interpret the quality and quality of the material. More broadly, the resulting PINN-based approach could be a promising approach for solving ill-posed inverse problems, often encountered in the nondestructive evaluation of materials.

Introduction

In recent years, the availability of large data sets, combined with sophisticated algorithms and an exponential growth in computational power, has led to unprecedented usage of data in machine learning techniques. Machine learning has become a major tool in a spectrum of disciplines, ranging from classification problems, including speech recognition, natural language processing, and computer vision, to complex regression problems like the approximation of nonlinear and discontinuous functions. However, the applications of neural networks are less explored in the engineering fields. Physics-informed machine learning approaches offer a new paradigm for handling physical laws with observational data. Recently, such machine learning-based techniques have obtained a lot of attention around the world, with (1) and the references therein. In particular, Raissi et al. (1) proposed the PINN methodology, which can accurately solve the forward problem of learning the

A zebrfish model for calcineurin-dependent brain function

Sara Tucker Edmister 1, Rahma Ibrahim 1, Rohit Kakodkar 2, Jill A Kreiling 1, Robbert Creton 3

Affiliations + expand

Free PMC article
DART
Data Assimilation Research Testbed

Cross-lab
Cross-institution
Cross-country
Cross-world
NCAR Real-time ensemble prediction system

Severe weather forecast for two days compared to NWS warnings

- WRF, 10 member ensemble, GFS for boundary conditions
- Continuous operation from April 2015 to December 2017
- 48 hour forecasts at 3km resolution
- First continuously cycling ensemble system for CONUS
- CISC Dedicated Queues and Computing Support were Vital

Advancing models & observations together

WRF-Hydro/DART: Florence 2018

Hurricane Florence made landfall near Wrightsville Beach, North Carolina at 7:16 a.m. ET September 14. The GOES East satellite captured this geocolor image at 7:45 a.m. ET.

Winds up to 150 mph (240 km/hr)
Damage: $24.23 billion
NOAA/NWS/NCEP/WPC

WRF-Hydro/DART: DA Impact

Assimilation happens every hour
Correction along major reaches. DA is adding water to the stream channels.
Better Scientific Software (BSSw)

Software—the foundation of discovery in computational science & engineering—faces increasing complexity in computational models and computer architectures. BSSw provides a central hub for the community to address pressing challenges in software productivity, quality, and sustainability.
Meet Our Fellows

The BSSw Fellowship program gives recognition and funding to leaders and advocates of high-quality scientific software. Meet the Fellows and Honorable Mentions and learn more about how they impact Better Scientific Software.

2023 Class

Nicole Brewer
Arizona State University
Improving accessibility of data and software with scientific web apps

Myra Cohen
Iowa State University
Techniques for scientific software testing

Johannes Doerfert
Lawrence Livermore National Laboratory
Demystifying the compiler black box

William Hart
Sandia National Laboratories
Best practices for software supply chain security

Helen Kershaw
National Center for Atmospheric Research

Rafael Mudafor
National Renewable Energy Laboratory

https://bssw.io/pages/meet-our-fellows
What problem am I trying to solve?
"I'm one of the few people you'll meet who's written more books than they've read."

GARTH MARENGHI

What outcomes would I like to see?
Outcomes

- People reviewing early and often
- People reviewing *each other's code*
- Comfortable with napkin explanations of code
- Become a *better reviewer*
- Better code
- Take a look inside
- More open source contributors!
Outcomes

● People reviewing early and often
● People reviewing each others code
● Comfortable with napkin explanations of code
● Become a better reviewer
● Better code
● Take a look inside
● More open source contributors!  

Ulterior Motive
Onboard new contributors to DART
Onboard new contributors to DART

But not be specific to DART
Code review is a skill
Learning *several things* at once

- The mechanics of git and GitHub
- A new programming language
- New science
- Culture of new team
Learning **several things** at once

- The mechanics of git and GitHub
- A new programming language
- New science
- Culture of new team
Learning several things at once

- The mechanics of git and GitHub
- A new programming language
- New science
- Culture of new team
- And code review

Seasoned professional
Early career
The Tutorial
code-review.org
The Tutorial

Three sets of exercises

● No code
● Python
● Fortran
No code exercises

- Cake recipe
- Article on the women’s world cup
- Origami instructions to make a fish

Text Exercise 1: Cake recipe

People have reported several problems with a recipe

Some examples of user feedback:

- Worst Chocolate Cake ever! Not even chocolatey
- Yuk! salty

Take a look at the recipe cake.md.

- Are there any problems?
The Tutorial

Three sets of exercises

- No code
- Python
- Fortran
The Tutorial

Three sets of exercises

- No code
- Python
- Fortran
The Tutorial

Three sets of exercises

- No code
- Python
- Fortran

- Issue + prompts
- Pull request + prompts
Setting up the tutorial on GitHub
Set up the Tutorial on GitHub

To do the tutorial exercises interactively you will need a GitHub account.

If you are not ready to set up a GitHub account and run workflows, a ‘take-a-look’ repository with the exercise issues and pull requests is available at take-a-look. You can follow along with the tutorial and read the issues and pull requests without having to run your own GitHub actions. Skip head to the exercises.

For those working through the exercises, this setup guide will take you through forking the tutorial, enabling issues, setting read and write permissions for workflows, and switching on workflows.

Fork the tutorial repository

The tutorial repository is github.com/scientific-software-reviewers/tutorial

![GitHub repository](image-url)

Welcome to code review tutorial GitHub repository

Uncheck Copy the main branch only and click Create fork

- Fork
- All branches
- Enable workflows
- ...
Set up the Tutorial on GitHub

To do the tutorial exercises interactively you will need a GitHub account.

If you are not ready to setup a GitHub account and run workflows, a ‘take-a-look’ repository with the exercise issues and pull requests is available at take-a-look. You can follow along with the tutorial and read the issues and pull requests without having to run your own GitHub actions. Skip head to the exercises.

For those working through the exercises, this setup guide will take you through forking the tutorial, enabling issues, setting read and write permissions for workflows, and switching on workflows.

Fork the tutorial repository

The tutorial repository is github.com/scientificsoftware-reviewers/tutorial

Barrier before I’ve started
take-a-look repository
Setting up the tutorial on GitHub
Set up the Tutorial on GitHub

To do the tutorial exercises interactively you will need a GitHub account.

If you are not ready to setup a GitHub account and run workflows, a ‘take-a-look’ repository with the exercise issues and pull requests is available at take-a-look. You can follow along with the tutorial and read the issues and pull requests without having to run your own GitHub actions. Skip ahead to the exercises.

For those working through the exercises, this setup guide will take you though forking the tutorial, enabling issues, setting read and write permissions for workflows, and switching on workflows.

Fork the tutorial repository

The tutorial repository is github.com/scientific-software-reviewers/tutorial
Create the exercises

- Fork
- All branches
- Enable workflows
- Run workflows
create_exercises
create_exercises.yaml

0 workflow runs

This workflow has a workflow_dispatch event trigger.

This workflow has no runs yet.
This workflow has a workflow_dispatch event trigger.

This workflow has no runs yet.
This workflow has a `workflow_dispatch` event trigger.

This workflow has no runs yet.
Workflow run was successfully requested.

**Actions**

- New workflow

**create_exercises**

*create_exercises.yaml*

**0 workflow runs**

---

**Event**  
**Status**  
**Branch**  
**Actor**

This workflow has a `workflow_dispatch` event trigger.

**create_exercises**

create_exercises #1: Manually run by hkershaw-brown

Run workflow
<table>
<thead>
<tr>
<th>Issue</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fortran: Exercise 2</td>
<td>Fortran</td>
</tr>
<tr>
<td>Python: Exercise 2</td>
<td>python</td>
</tr>
<tr>
<td>Python: Exercise 1</td>
<td>python</td>
</tr>
<tr>
<td>Text: Exercise 2</td>
<td>text</td>
</tr>
<tr>
<td>Text: Exercise 1</td>
<td>text</td>
</tr>
<tr>
<td>Text: Exercise 3</td>
<td>text</td>
</tr>
</tbody>
</table>

ProTip! Adding `nolabel` will show everything without a label.
Navigating the exercises
Issues

Pull Requests
Issues

Experiencing Issues

An issue is a way to discuss, plan and track work on a GitHub repository.

Issues can be bugs, complaints from users, requests for new features or added functionality.

When reading through an issue,

- Are there multiple problems reported in the issue?
- Can you confirm the issue by reading the code or documentation?
- Do you need to run the code to confirm the issue?
- Can you reproduce the problem?

Each tutorial exercise has an issue describing the problem. This issue contains a link to the code under discussion, and some questions to think about when looking at the issue, the code, and the pull request.

Bonus points:

Think about how you would like people to report issues with your own code.

Would you use GitHub issues templates to prompt people to provide relevant information?

What is important information you would like someone to give in an issue?

- version of the code being used?
- a small example the shows the bug?
- screenshots of the problem?
- error messages?
- desired solution?
- operating system where the problem occurred (Windows, Mac, Linux)?
Pull Requests
Pull Requests
Navigating Pull Requests

**size and scope**
Adding suggestions

Suggestions are the same as comments, but you suggest an edit to the code that can be committed from the pull request. Click the suggestion icon in the comment box.

The lines you have selected will show up. Edit this with what you think should be there. You can click preview to see your code changes.

Try committing changes from a suggestion.

Add your review

When you’re ready to add your review click the green Reviewer changes button.
Adding your review
Reviewing
Reviewing

Being reviewed
Does the pull request address the issue?

Are there any deal breakers that would stop you accepting the changes?

Can you suggest any improvements?

What is a good way to phrase your suggested improvements?

Is the solution overly complicated? Are the comments up to date, necessary, helpful?

Would you accept the pull request as it is now? Are your suggested changes must-do? nice-to-have? nitpicks?

How would you communicate this?

Do you spend a lot of time reviewing the code style? Is it worth having a style guide for contributors? Can you make use of an existing style guide? Or a linter?
Reviewing

Does the pull request address the issue?

Are there any deal breakers that would stop you accepting the changes?

Can you suggest any improvements?

What is a good way to phrase your suggested improvements?

Is the solution overly complicated? Are the comments up to date, necessary, helpful?

Would you accept the pull request as it is now? Are your suggested changes must-do? nice-to-have? nitpicks?

How would you communicate this?

Do you spend a lot of time reviewing the code style? Is it worth having a style guide for contributors? Can you make use of an existing style guide? Or a linter?

---

Being reviewed

When putting in a pull request, how can you make it easy for a reviewer to understand what you have done?

What makes a good pull request, what makes a bad pull request?

Can you commit code in a way that lets someone review your code more easily? Should you separate functional changes from style changes?

Would you use a tool such as commitizen to prompt yourself at commit time? Why? Why not?
Mechanics of the tutorial
Adding exercises
Adding exercises

Two GitHub workflows:

`create_exercises` `create_exercises.yaml`

`reset_exercises` `close_issues_and_pulls.yaml`

https://github.com/scientific-software-reviewers/tutorial
Adding exercises

<table>
<thead>
<tr>
<th>Directory</th>
<th>Content</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>.github/workflows</td>
<td>code review tutorial</td>
<td>3 months ago</td>
</tr>
<tr>
<td>Fortran</td>
<td>code review tutorial</td>
<td>3 months ago</td>
</tr>
<tr>
<td>issues</td>
<td>code review tutorial</td>
<td>3 months ago</td>
</tr>
<tr>
<td>pull_requests</td>
<td>code review tutorial</td>
<td>3 months ago</td>
</tr>
<tr>
<td>python</td>
<td>code review tutorial</td>
<td>3 months ago</td>
</tr>
<tr>
<td>text</td>
<td>code review tutorial</td>
<td>3 months ago</td>
</tr>
<tr>
<td>.gitignore</td>
<td>code review tutorial</td>
<td>3 months ago</td>
</tr>
<tr>
<td>LICENSE</td>
<td>Initial commit</td>
<td>10 months ago</td>
</tr>
<tr>
<td>README.md</td>
<td>code review tutorial</td>
<td>3 months ago</td>
</tr>
</tbody>
</table>
Adding exercises

issues/{Language}-ex#{Language}-issue.md

pull_requests/{Language}-ex#{Language}-pull_body.md

Branch: {Language}-#{Language}
Adding exercises

issues/{Language}-ex{#}-issue.md

pull_requests/{Language}-ex{#}-pull_body.md

Branch: {Language}-{#}

.github/workflows/create_exercises.yaml is the action that takes ‘Language’, and for each exercise {1..n}:
Adding exercises

issues/{{Language}}-ex{#}-issue.md

pull_requests/{{Language}}-ex{#}-pull_body.md

Branch: {{Language}}-{#}

.github/workflows/create_exercises.yaml is the action that takes ‘{{Language}}’, and for each exercise {1..n}:

- creates any issues {{Language}}-{1…n}. 
Adding exercises

issues/{Language}-ex{#}-issue.md

pull_requests/{Language}-ex{#}-pull_body.md

Branch: {Language}-{#}

github/workflows/create_exercises.yaml is the action that takes ‘Language’, and for each exercise {1..n}:

- creates any issues {Language}-{1…n}.
- creates pull requests {1..n} for branches {Language}-{1..n} using text from {Language}-pull_body.md
Adding exercises

issues/{Language}-ex{#}-issue.md

pull_requests/{Language}-ex{#}-pull_body.md

Branch: {Language}-{#}

.github/workflows/create_exercises.yaml is the action that takes ‘Language’, and for each exercise {1..n}:

- creates any issues {Language}-{1…n}.
- creates pull requests {1..n} for branches {Language}-{1..n} using text from {Language}-pull_body.md

Code is in the directories:

{Language}/exercise{#}
Adding exercises

.github/workflows/close_issues_and_pulls.yaml

Resets the exercises:
Roll back the repo with git reset hard
Restores the `{Language}`-{#} branch from a corresponding `backup-{Language}`-{#} branch
Squashing git history

main branch has only two commits:

- Initial commit
- Code review tutorial
Squashing git history

main branch has only two commits:

- Initial commit
- Code review tutorial

Examine commit history in exercises
Code Review
Tutorial Blog Get Involved Search docs...

Tutorial Exercises

There are three sets of exercises, all on GitHub. Follow the setup instructions to create your own tutorial repository to work on.

If you are not ready to setup a GitHub account and run workflows, a “take-a-look” repository with the exercise issues and pull requests is available at take-a-look. You can follow along with the tutorial and read the issues and pull requests without having to run your own GitHub actions.

For those working through the exercises, running the create_exercises workflow will generate the test, python, or Fortran exercises in your fork of the tutorial.

The exercises are in no particular order. You can start with whichever seems the most interesting/applicable to you. Each exercise consists of a GitHub issue describing a problem and a pull request which is a proposed solution to the issue. Your mission is to review the proposed solution.

An issue may be a complaint from a user, a bug report, a request for a new feature. Pull requests may be from a colleague, or if your code is publicly available, from someone who uses the code.

Code Review

https://github.com/scientific-software-reviewers/tutorial
Software Engineering is
“programming integrated over time”

Winters, T., Manshreck, T., & Wright, H. (2020). Software engineering at google: Lessons learned from programming over time. O'Reilly Media
Experiences from UCAR SEA
Software Engineering Assembly
What is the SEA?
UCAR Software Engineering Assembly

- Foster community for software engineering professionals within UCAR
- Facilitate effective participation
- Advocate for Software Engineers

SEA Improving Scientific Software Conference April 15th-18th
https://sea.ucar.edu/conference/2024
Code Review
Experiences from UCAR SEA
Software Engineering Assembly
Experiences from UCAR SEA

Join the UCAR Software Engineering Assembly for a lightly-moderated discussion on code review across UCAR.

Any discussion topics are welcome, as are all experience levels. We encourage you to share your good and bad experiences with code review.

- Do you use code review in your group? Who does the reviewing? Have you used code review to transfer knowledge between team members?
- Reviewing is hard. Being reviewed can be difficult. How do you give and receive constructive and actionable criticism?
- Do you do in person code reviews? Offline code-reviews? What works, what doesn’t?
- Do you spend too much time in review, and have ideas to improve the process?
Experiences from UCAR SEA

Code review feels like someone works with me and we learn from each other

GitHub made it much easier to code review.

When people do not know much about what others do in the code, review gives an opportunity to learn about what is going on in the project

Downside: Back/forth that happens, especially since the code review is not #1 priority. Can slow down the process.

Getting very burned out with code reviews generally e.g. Do a review, wait ~2 weeks, can feel really negative sometimes
Recently got more negative on it but would love to hear positive experiences about it
Used to do code reviews in person years ago. Finding bugs and avoiding problems down the line works great. Can’t imagine deploying code without reviews. Couldn’t maintain the code without reviews.

Experience mostly getting my code being reviewed rather than reviewing others’. Need to coordinate with each other to find the time. Trick is that it’d be helpful to walk the reviewer through the code first.

The objectives can be communicated well beforehand using a pull request template to reduce the overhead of back & forth and expectations for a due date for the pull request can be set.

Communicating what to look at in the code is really important.
Experiences from UCAR SEA

A lot of friction points about code review. Ethics around code review is not clear. Code review is a lot of times not equitable, e.g. more pushback for women’s code.

Code style actions, automation could be helpful with the code reviewing process to reduce unwanted reviewing (code styling, etc.)

Systemic Gender Inequities in Who Reviews Code
The Pushback Effects of Race, Ethnicity, Gender, and Age in Code Review – Communications of the ACM
Presentation by Dr. Kelly Blincoe about code review as a socio-technical activity. Includes relevant data and potential policy implications on code review processes and impact.

Pick the most impactful aspects of the code to comment on, no need to mention everything. Impact can include functionality, quality, maintainability, readability, testability.
Experiences from UCAR SEA

Submitting changes without sufficient descriptions is less helpful.

Sometimes reviews have a lot of back & forth, and can get political. Try to keep it very non-personal. The thing being reviewed is not the person but the code that will benefit an entire project/organization.

It’s a joint responsibility.

Encourage "the code" and not "your code". We are not our code.

Make it clear about the asynchronous aspect of the PRs. Also use “why would you do that?” for asking the reasoning (?)

Having been in both scientist and developer perspectives, set expectations and convey what the goals are for each group, collective set of expectations. And, things may differ from person to person, even if they are all one kind (e.g. scientist).

Consistency. Type of code you are working on (pure research vs. operational product/deliverable) and how you set expectations is also very important.
Experiences from UCAR SEA

1:1 code review in person is a bit different than remote.

When getting someone new to our code contributions, reach out individually with an email that clarifies some important points about the process.

Code review as an onboarding task

Do onboarding by working side-by-side rather than a remote pull request review process. Some form of pair programming.

How Microsoft do code reviews mentions the use of emojis to describe things like nitpick, thinking out loud, take it or leave it, etc.
Finding Community
Finding Community

- **US-RSE.** A community of people who make research software happen.
- Society of Research Software Engineering which emerged from the successful grass-roots RSE movement and is the successor to the UK RSE Association.
- **Better Scientific Software.** A hub for scientific software development resources.
- **Campus Champions.** Uniting Research Computing Facilitators
- ...
Finding Community

- **US-RSE.** A community of people who make research software happen.
- Society of Research Software Engineering which emerged from the successful grass-roots RSE movement and is the successor to the UK RSE Association.
- **Better Scientific Software.** A hub for scientific software development resources.
- **Campus Champions.** Uniting Research Computing Facilitators
- …

hkershaw@ucar.edu  
USRSE slack

code-review.org