

SOFTWARE METAPAPER

PFHub: The Phase-Field Community Hub

Daniel Wheeler¹, Trevor Keller¹, Stephen J. DeWitt², Andrea M. Jokisaari³, Daniel Schwen³, Jonathan E. Guyer¹, Larry K. Aagesen³, Olle G. Heinonen⁴, Michael R. Tonks⁵, Peter W. Voorhees⁶ and James A. Warren⁷

¹ Materials Science and Engineering Division, Material Measurement Laboratory, National Institute of Standards and Technology, Gaithersburg, MD, US

² Materials Science and Engineering Department, University of Michigan, Ann Arbor, MI, US

³ Fuel Modeling and Simulation Department, Idaho National Laboratory, Idaho Falls, ID, US

⁴ Argonne National Laboratory, Lemont, IL, US

⁵ Department of Materials Science and Engineering, University of Florida, Gainesville, FL, US

⁶ Department of Materials Science and Engineering, Northwestern University, Evanston, IL, US

⁷ Material Measurement Laboratory Office, Material Measurement Laboratory, National Institute of Standards and Technology, Gaithersburg, MD, US

Corresponding author: Daniel Wheeler (daniel.wheeler@nist.gov)

Scientific communities struggle with the challenge of effectively and efficiently sharing content and data. An online portal provides a valuable space for scientific communities to discuss challenges and collate scientific results. Examples of such portals include the Micromagnetic Modeling Group (μ MAG) [1], the Interatomic Potentials Repository (IPR) [2, 3] and on a larger scale the NIH Genetic Sequence Database (GenBank) [4]. In this work, we present a description of a generic web portal that leverages existing online services to provide a framework that may be adopted by other small scientific communities. The first deployment of the PFHub framework supports phase-field practitioners and code developers participating in an effort to improve quality assurance for phase-field codes.

Keywords: phase-field; materials-science; jekyll-website; reproducible-science; scientific-portal

Funding statement: D.W. wishes to acknowledge the Materials Genome Initiative funding allocated to the National Institute of Standards and Technology. S.J.D wishes to acknowledge funding from the U.S. Department of Energy, Office of Basic Energy Sciences, Division of Materials Sciences and Engineering under Award #DE-SC0008637 as part of the Center for Predictive Integrated Structural Materials Science (PRISMS Center) at University of Michigan. P.W.V. is grateful for the financial assistance under the award 70NANB14H012 from the National Institute of Standards and Technology as part of the Center for Hierarchical Materials Design (CHiMaD).

(1) Overview

Introduction

Generally, small scientific communities do not have the resources to build and host dedicated web infrastructure to support their varied content and data requirements. In particular, hosting and supporting a complex content management system (CMS) including web servers, web frameworks and databases requires a great deal of configuration and long term support and funding. Furthermore, a turnkey CMS solution may not meet requirements for most scientific communities that often use arcane data formats and require custom data displays along with client-side automation. The PFHub effort, instead of focusing on the CMS tool, focuses on customizing and delivering the client-side requirements whilst delegating back-end functionality to external services that provide dependable APIs [5].

The phase-field method (PFM) describes material interfaces at the mesoscopic scale between atomic scale models and macroscale models [6]. The PFM is well established and there are an assortment of code frameworks (*e.g.*, FiPy [7], MMSP [8], MOOSE [9], PRISMS-PF [10]) available for solving the wide variety of phenomena associated with phase-field (*e.g.* dendritic growth, spinodal decomposition, grain growth) [11]. However, it is difficult for novice as well as seasoned phase-field practitioners to assess the capability of codes for different phenomena without extensive prototyping and groundwork. PFHub aims to provide a low barrier for comparing code output data using a standard set of metrics.

PFHub is a community effort spearheaded by the Center for Hierarchical Materials Design at Northwestern University and the National Institute of Standards and Technology in support of phase-field code development.

The current PFHub deployment [12] focuses on improving cross-collaboration between phase-field code developers and practitioners by providing a standardized set of benchmark problems [13, 14] and a workflow for uploading and comparing benchmark results from different phase-field codes.

Community based scientific efforts often require web services to share and display data in unique ways between groups and institutions. These services are difficult to implement due to the groundwork required to investigate and prototype the many data-sharing and CMS tools available. The PFHub framework provides a template for other scientific projects beyond the phase-field community. The method outlined in this paper of using static infrastructure coupled with small independent third party web services provides a flexible approach eliminating the initial prototyping and on-going maintenance required for new infrastructure, while allowing developers to focus on their unique front-end data views.

This paper presents the first deployment of the PFHub framework including its client-side focused design, how it

employs external services and metadata about the code base. The paper describes the relative ease with which other scientific groups might adapt the framework for their own purposes and deploy using the fully reproducible Nix environment [15].

Implementation and architecture

The PFHub framework provides a template for other small scientific communities to host custom content and integrate data from members of their community. The current deployment (see **Figure 1**) provides a facility for uploading, displaying and comparing results from benchmark problems supporting phase-field code developers and practitioners. However, the framework and overall philosophy are broadly transferable to other communities with some custom configuration and content generation. The framework uses the Jekyll static website generator [16] along with automated front-end processing to eliminate the need for a CMS [5], which is generally costly to maintain especially for small scientific communities with limited funding and staffing. The framework relies

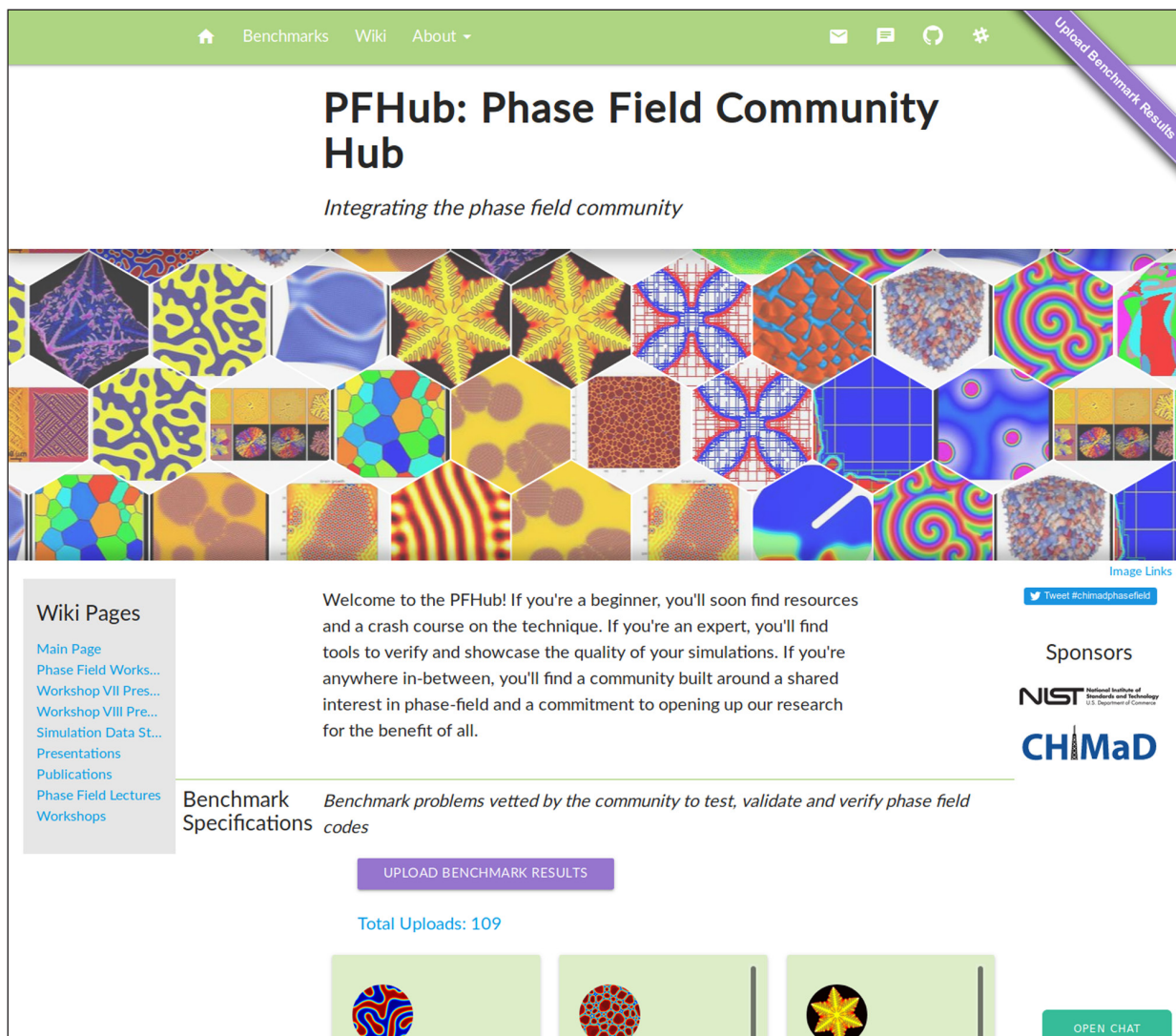


Figure 1: Front page of the current version of the deployed PFHub website showing links to the wiki and upload pages. The hexbin splash provides links to phase-field papers and websites. This can easily be modified with alternative links and images by editing a YAML file.

on the API, WebSocket and webhook infrastructure that underpins the modern web and allows external services to have full-duplex communication between servers and browsers. In particular, PFHub relies on GitHub's well maintained API and webhook functionality for external services (such as Travis CI [17] and Staticman [18]).

The workflow for uploading benchmark results relies on third party tools using the following steps, illustrated in **Figure 2**.

1. The users are first required to upload simulation outputs to an archival resource (*e.g.*, Figshare¹ [19]) configured with permissive cross-origin resource sharing (CORS).
2. The metadata summarizing each simulation is entered into a form on the website (see **Figure 3**), including relevant details such as memory usage, run time and links to the data archived in the first step.
3. Upon submission, the Staticman app [18] submits the entered metadata as a pull request against the PFHub repository hosted by GitHub. The metadata is stored in a YAML file with a unique path in the repository, see [20] for an example.
4. Travis CI [17] performs linting on the submission and then launches a temporary version of the proposed website using Surge [21]. The PFHub admins can then examine the new submission and further changes can be made if necessary.
5. Once review has been completed to the satisfaction of both the uploading scientist and the website maintainers, the pull request is merged and served to the World Wide Web using a hosting service compatible with GitHub Pages.

A combination of Jekyll templates and CoffeeScript are used to access and download the data links in the submitted

YAML files and then display the data in interactive plots on the website. CoffeeScript is a higher level language than JavaScript and, thus, more readable whilst also allowing programming in a more functional manner which makes data manipulation pipelines both more succinct and easier to understand. The interactive plots (see **Figure 4**) are displayed using the Plotly JavaScript Graphing Library [22] as it provides a programmable interface and requires minimal configuration, see [20] for typical data displayed on the interactive plots.

The current deployment of PFHub has benchmark specifications consisting of equations, narrative, plots and code samples, and are composed in Jupyter Notebooks. The Jupyter Notebooks are included as static objects in the website after translation into HTML using the nbconvert tool [23]. There are currently 7 benchmark problems each with a number of variations. At the time of writing there are 109 separate benchmark result uploads [12] submitted as pull requests and approved following code review to ensure compatibility with the website.

The combination of a central repository on GitHub for website source code and metadata with distributed data records on third-party archives avoids the complexity and administrative overhead of maintaining a live database and associated back-end application.

Quality control

The framework has a fully automated test recipe deployed on Travis CI with an environment built using the Nix Package Manager [15]. A fully automated test environment using continuous integration allows all developers and users to have common feedback on code updates and determine the compatibility of result uploads with the deployed website. The environment is pinned to a specific version of the Nix Packages Collection (Nixpkgs) [24], ensuring fully reproducible

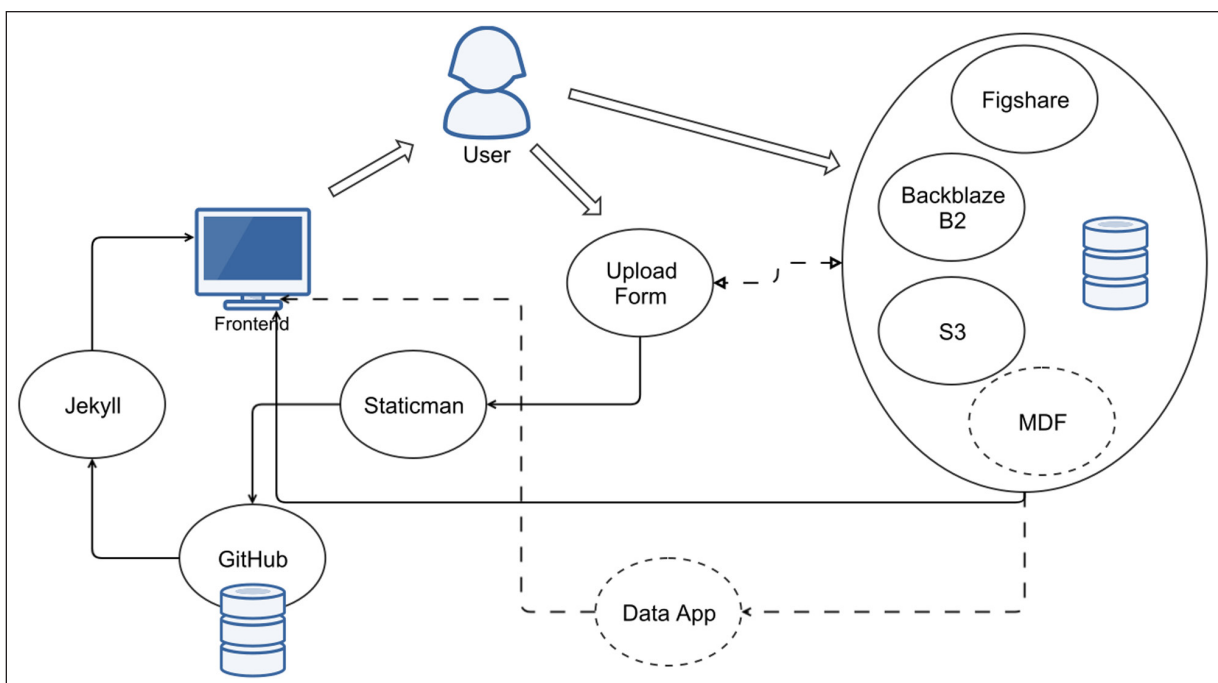


Figure 2: Schematic overview of the PFHub framework for building scientific research portals, simply.

Figure 3: The form used to enter simulation result metadata and then upload to the repository via a GitHub pull request allowing the data to be checked before pushing to the website.

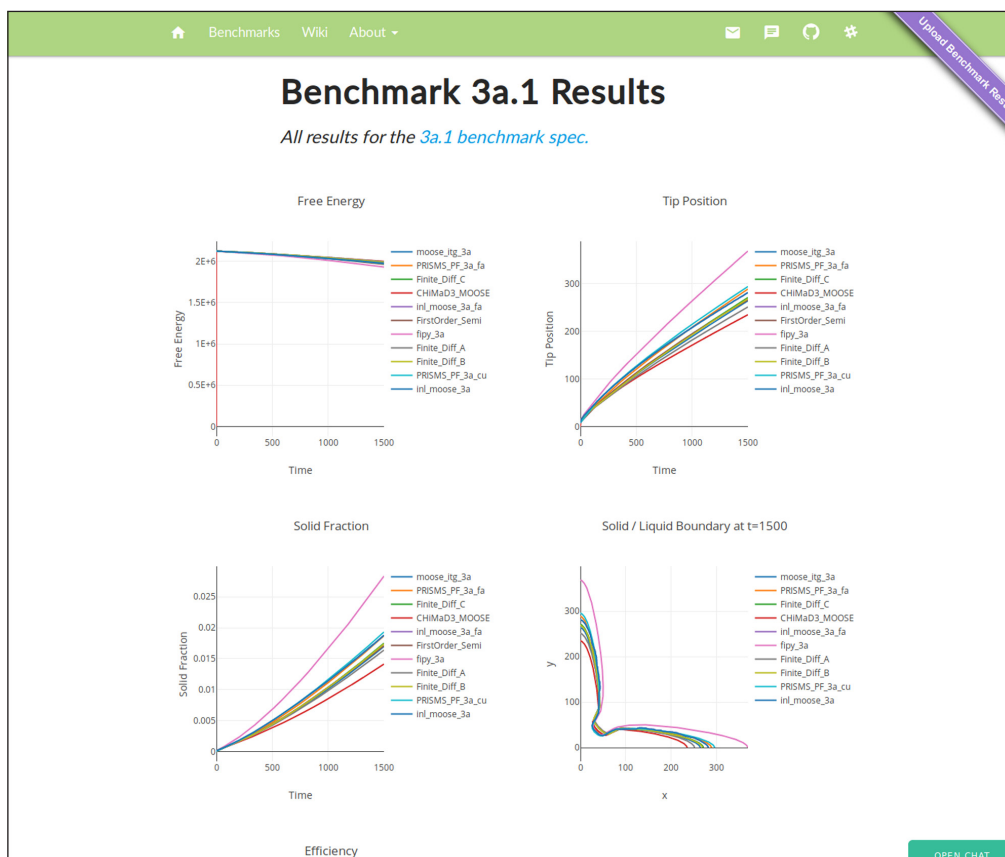


Figure 4: Results comparison page for benchmark 3 (Dendritic Growth). The data for each upload is plotted asynchronously and the page is not affected if a data link dies.

build and test phases as well as ensuring that the development and automated testing environments are identical. The full test recipe is outlined in a YAML file, `.travis.yml`, stored in the repository [25] and consists of the following steps.

1. Build the Nix environment from a persistent cache on Travis CI, reducing the build time.
2. Run automated tests on Jupyter Notebooks using NBval [26] and Py.test [27].
3. Run validation tests on HTML files using HTML-Proofer [28].
4. Lint and test front-end CoffeeScript using Coffeelint [29] and Mocha [30], respectively.
5. Display a temporary version of the website using Surge [21] for visual review.

(2) Availability

Operating system

The PFHub framework can be deployed on any platform supporting Nix, which includes all contemporary Linux and macOS platforms. Since the framework is built with Jekyll and automated front-end processing, it can be deployed on GitHub's Pages infrastructure, which enables streamlined deployment without the need for any back-end infrastructure and, thus, is largely platform independent. For development purposes, a local installation of either Nix (on Linux or Mac) or Docker (on Linux, Mac or Windows) is required.

Programming language

PFHub is currently built and tested using the programming languages and versions outlined in **Table 1**.

Additional system requirements

There are no additional system requirements.

Dependencies

The entire environment can be built using the Nix Package Manager so the only required dependency is a functional Nix installation. The PFHub framework has over 2000 separate package dependencies using data from the Nix package manager. The full dependency graph for PFHub can be seen online [31].

List of contributors

This list is for contributors to the code base, but not those that have only uploaded output results to the website.

Table 1: PFHub programming languages and corresponding supported versions.

Language	Version
HTML	5
Jupyter Notebook	5.4.0
JavaScript	5
Nix	2.1.3
CoffeeScript	1.12.7
CSS	4

1. Wheeler, Daniel; @wd15
2. Keller, Trevor; @tkphd
3. DeWitt, Stephen J.; @stvdwtt
4. Jokisaari, Andrea M.; @amjokisaari
5. Schwen, Daniel; @dschwen
6. Guyer, Jonathan E.; @guyer

Also, see the contributors list on GitHub [32].

Software location

Archive

Name: Zenodo

Persistent identifier: 10.5281/zenodo.2592705

Licence: NIST Software License [33]

Publisher: Daniel Wheeler

Version published: v0.1

Date published: 13/03/19

Code repository

Name: GitHub

Persistent identifier: <https://github.com/usnistgov/pfhub/tree/v0.1>

Licence: NIST Software License [33]

Date published: 13/03/19

Languages

English

(3) Reuse potential

The PFHub framework can be readily adopted by other communities that want to follow a CMS-free philosophy and use well supported external services. The website infrastructure can be cloned as a Git repository or downloaded as a ZIP archive and deployed with minimum effort. The mechanism for uploading data using Staticman can be easily configured for a new repository location. However, customizing the content of the website for a particular scientific community would require considerable effort. The current effort is closely integrated with GitHub, but future deployments could be modified to use other repository services such as GitLab or BitBucket.

The following steps are the more challenging aspects of deploying the framework for a new community.

- Upload new data upload specifications (*e.g.* the phase-field benchmarks in the PFHub website [12]) in a format that Jekyll can parse, *e.g.*, Jupyter Notebook, Markdown or HTML.
- Edit the `benchmarks.yml` file to reflect the new upload requirements and describe the figures that need to be generated on the upload comparison pages.
- Edit the `_config.yml` file to update links and text related to the configuration for all aspects of the website.
- Update Markdown files to reflect the new content and mission of the scientific community.
- Remove data and files that are not required by the new community.

Further details on deployment and development of PFHub can be found in the development guide [34].

Currently, a deployment for a new community has not been attempted and, thus, the above steps need to be refined and documented.

Note

¹ Certain commercial equipment, instruments, or materials (or suppliers, or software, ...) are identified in this paper to foster understanding. Such identification does not imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the materials or equipment identified are necessarily the best available for the purpose.

Acknowledgements

We gratefully acknowledge input and guidance from all participants in the series of Phase-Field workshops held between 2015 and 2018 at the Center for Hierarchical Material Design [35].

Competing Interests

The authors have no competing interests to declare.

References


1. *μMAG: The Micromagnetic Modeling Activity Group*. 2019. URL: <https://www.ctcms.nist.gov/~rdm/mumag.org.html> (visited on 03/11/2019).
2. **Hale, L M, Trautt, Z T and Becker, C A** July 2018 “Evaluating variability with atomistic simulations: the effect of potential and calculation methodology on the modeling of lattice and elastic constants”. en. In: *Modelling and Simulation in Materials Science and Engineering*, 26(5): 055003. ISSN: 0965-0393, 1361-651X. DOI: <https://doi.org/10.1088/1361-651X/aabc05>
3. **Becker, C A**, et al. Dec. 2013 “Considerations for choosing and using force fields and interatomic potentials in materials science and engineering”. en. In: *Current Opinion in Solid State and Materials Science*, 17(6): 277–283. ISSN: 13590286. DOI: <https://doi.org/10.1016/j.cossms.2013.10.001>
4. *GenBank: NIH genetic sequence database*. URL: <https://www.ncbi.nlm.nih.gov/genbank/> (visited on 03/26/2019).
5. **Cole, D** *How We Build CMS-Free Websites*. July 2018. URL: <https://medium.com/devseed/how-we-build-cms-free-websites-d7e19d94a0ff> (visited on 03/11/2019).
6. **Moelans, N, Blanpain, B and Wollants, P** 2008 “An introduction to phase-field modeling of microstructure evolution”. In: *Calphad*, 32(2): 268–294. ISSN: 0364-5916. DOI: <https://doi.org/10.1016/j.calphad.2007.11.003>
7. **Guyer, J E, Wheeler, D and Warren, J A** 2009 “FiPy: Partial Differential Equations with Python”. In: *Computing in Science & Engineering*, 11(3): 6–15. ISSN: 1521-9615. DOI: <https://doi.org/10.1109/MCSE.2009.52>
8. **Gruber, J**, et al. Mar. 2019 *mesoscale/mmsp: Zenodo integration*. DOI: <https://doi.org/10.5281/zenodo.2583258>
9. **Tonks, M R**, et al. 2012 “An object-oriented finite element framework for multiphysics phase field simulations”. In: *Computational Materials Science*, 51(1): 20–29. ISSN: 0927-0256. DOI: <https://doi.org/10.1016/j.commatsci.2011.07.028>
10. **DeWitt, S**, et al. Mar. 2019 *prisms-center/phaseField: PRISMS-PF (Version 2.1.1)*. DOI: <https://doi.org/10.5281/zenodo.2583308>
11. **Tonks, M R and Aagesen, L K** 2019 “The Phase Field Method: Mesoscale Simulation Aiding Material Discovery”. In: *Annual Review of Materials Research*, 49(1): 79–102. DOI: <https://doi.org/10.1146/annurev-matsci-070218-010151>
12. *PFHub: The Phase Field Community Hub*. URL: <https://pages.nist.gov/pfhub> (visited on 03/27/2019).
13. **Jokisaari, A M**, et al. 2017 “Benchmark problems for numerical implementations of phase field models”. In: *Computational Materials Science*, 126: 139–151. ISSN: 0927-0256. DOI: <https://doi.org/10.1016/j.commatsci.2016.09.022>
14. **Jokisaari, A M**, et al. 2018 “Phase field benchmark problems for dendritic growth and linear elasticity”. In: *Computational Materials Science*, 149: 336–347. ISSN: 0927-0256. DOI: <https://doi.org/10.1016/j.commatsci.2018.03.015>
15. *Nix Package Manager*. URL: <https://nixos.org/nix/> (visited on 03/13/2019).
16. *Jekyll*. URL: <https://jekyllrb.com/> (visited on 03/14/2019).
17. *Travis CI: Test and Deploy with Confidence*. URL: <https://travis-ci.org/> (visited on 03/11/2019).
18. *Staticman: Static sites with superpowers*. URL: <https://staticman.net> (visited on 03/11/2019).
19. *Figshare*. URL: <https://figshare.com/> (visited on 03/27/2019).
20. *Typical PFHub Data*. URL: https://github.com/usnistgov/pfhub/blob/master/_data/simulations/fipy_3a/meta.yaml (visited on 08/06/2019).
21. *Surge: Static web publishing for Front-End Developers*. URL: <https://surge.sh/> (visited on 03/11/2019).
22. Plotly Technologies Inc. *Collaborative data science*. 2015. URL: <https://plot.ly> (visited on 03/20/2019).
23. **Kluyver, T** et al. 2016 “Jupyter Notebooks – a publishing format for reproducible computational workflows”. In: *Positioning and Power in Academic Publishing: Players, Agents and Agendas*. Ed. by F. Loizides and B. Schmidt, 87–90. IOS Press. DOI: <https://doi.org/10.3233/978-1-61499-649-1-87>
24. *Nix Packages Collection*. URL: <https://github.com/NixOS/nixpkgs> (visited on 04/05/2019).
25. *Travis CI Recipe for PFHub*. URL: <https://github.com/usnistgov/pfhub/blob/master/.travis.yml> (visited on 03/13/2019).
26. *NBVal: Py.test plugin for validating Jupyter notebooks*. URL: <https://github.com/computationalmodelling/nbval> (visited on 03/14/2019).
27. *Py.test*. URL: <https://docs.pytest.org/en/latest/> (visited on 03/14/2019).
28. *HTML Proofer*. URL: <https://github.com/gjtorikian/html-proofer> (visited on 03/14/2019).

29. *CoffeeLint*. URL: <https://github.com/clutchski/coffeelint> (visited on 08/05/2019).
30. *Mocha*. URL: <https://mochajs.org/> (visited on 08/05/2019).
31. *PFHub Dependency Graph*. URL: <https://github.com/usnistgov/pfhub/wiki/PFHub-Dependency-Graph> (visited on 08/06/2019).
32. *List of PFHub Contributors*. URL: <https://github.com/usnistgov/pfhub/graphs/contributors> (visited on 03/13/2019).
33. *NIST Software License*. URL: <https://www.nist.gov/director/copyright-fair-use-and-licensing-statements-srd-data-and-software> (visited on 03/13/2019).
34. *PFHub Development Guide*. URL: <https://pages.nist.gov/pfhub/DEVELOPMENT/> (visited on 08/06/2019).
35. *CHiMaD Phase-Field Workshops*. URL: <https://pages.nist.gov/pfhub/wiki/workshops/> (visited on 03/14/2019).

How to cite this article: Wheeler, D, Keller, T, DeWitt, S J, Jokisaari, A M, Schwen, D, Guyer, J E, Aagesen, L K, Heinonen, O G, Tonks, M R, Voorhees, P W and Warren, J A 2019 PFHub: The Phase-Field Community Hub. *Journal of Open Research Software*, 7: 29. DOI: <https://doi.org/10.5334/jors.276>

Submitted: 17 April 2019 **Accepted:** 04 September 2019 **Published:** 24 September 2019

Copyright: © 2018 The Author(s). This is an open-access article distributed under the terms of the Creative Commons Attribution 4.0 International License (CC-BY 4.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited. See <http://creativecommons.org/licenses/by/4.0/>.

 *Journal of Open Research Software* is a peer-reviewed open access journal published by Ubiquity Press

OPEN ACCESS 