hIPPYlib Documentation

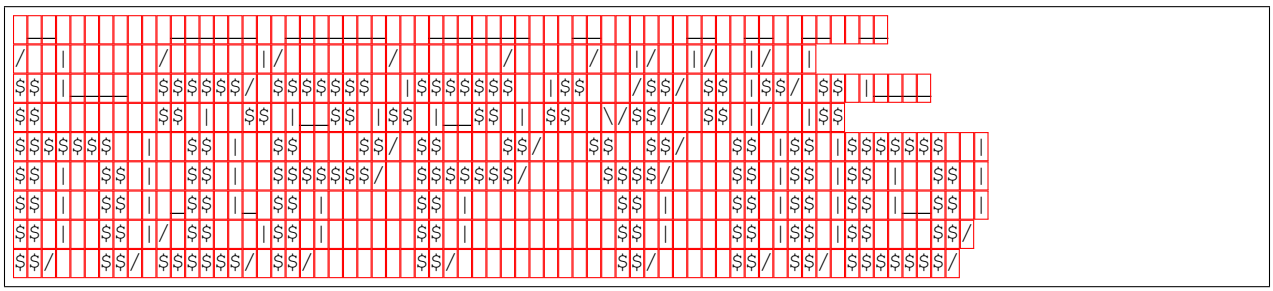
Release 3.0.0

Umberto Villa, Noemi Petra, Omar Ghattas

Nov 05, 2020

Contents

Inverse Problem PYthon library



https://hippylib.github.io

hIPPYlib implements state-of-the-art scalable algorithms for deterministic and Bayesian inverse problems governed by partial differential equations (PDEs). It builds on FEniCS (a parallel finite element element library) for the discretization of the PDE and on PETSc for scalable and efficient linear algebra operations and solvers.

For building instructions, see the file INSTALL.md. Copyright information and licensing restrictions can be found in the file COPYRIGHT.

The best starting point for new users interested in hIPPYlib's features are the interactive tutorials in the tutorial folder.

Conceptually, hIPPYlib can be viewed as a toolbox that provides the building blocks for experimenting new ideas and developing scalable algorithms for PDE-constrained deterministic and Bayesian inverse problems.

In hIPPYlib the user can express the forward PDE and the likelihood in weak form using the friendly, compact, near-mathematical notation of FEniCS, which will then automatically generate efficient code for the discretization. Linear and nonlinear, and stationary and time-dependent PDEs are supported in hIPPYlib. For stationary problems, gradient and Hessian information can be automatically generated by hIPPYlib using FEniCS symbolic differentiation of the relevant weak forms. For time-dependent problems, instead, symbolic differentiation can only be used for the spatial terms, and the contribution to gradients and Hessians arising from the time dynamics needs to be provided by the user.

Noise and prior covariance operators are modeled as inverses of elliptic differential operators allowing us to build on existing fast multigrid solvers for elliptic operators without explicitly constructing the dense covariance operator.

The key property of the algorithms underlying hIPPYlib is that solution of the deterministic and Bayesian inverse problem is computed at a cost, measured in forward PDE solves, that is independent of the parameter dimension.

hIPPYlib provides a robust implementation of the inexact Newton-conjugate gradient algorithm to compute the maximum a posterior (MAP) point. The gradient and Hessian actions are computed via their weak form specification in FEniCS by constraining the state and adjoint variables to satisfy the forward and adjoint problem. The Newton system is solved inexactly by early termination of CG iterations via Eisenstat-Walker (to prevent oversolving) and Steihaug (to avoid negative curvature) criteria. Two globalization techniques are available to the user: Armijo backtracking line search and trust region.

In hIPPYlib, the posterior covariance is approximated by the inverse of the Hessian of the negative log posterior evaluated at the MAP point. This Gaussian approximation is exact when the parameter-to-observable map is linear; otherwise, its logarithm agrees to two derivatives with the log posterior at the MAP point, and thus it can serve as a

proposal for Hessian-based Markov chain Monte Carlo (MCMC) methods. hIPPYlib makes the construction of the posterior covariance tractable by invoking a low-rank approximation of the Hessian of the log likelihood.

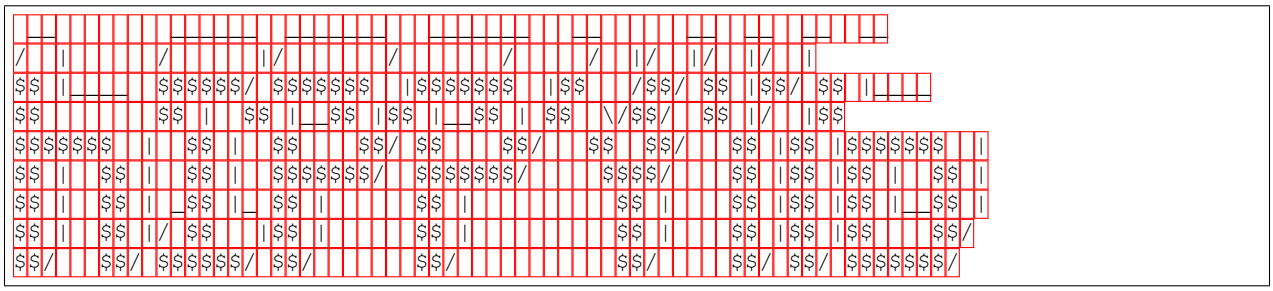
hIPPYlib also offers scalable methods for sample generation. To sample large scale spatially correlated Gaussian random fields from the prior distribution, hIPPYlib implements a new method that strongly relies on the structure of the covariance operator defined as the inverse of a differential operator: by exploiting the assembly procedure of finite element matrices hIPPYlib constructs a sparse Cholesky-like rectangular decomposition of the precision operator. To sample from a local Gaussian approximation to the posterior (such as at the MAP point) hIPPYlib exploits the low rank factorization of the Hessian of the log likelihood to correct samples from the prior distribution. Finally, to explore the posterior distribution, hIPPYlib implements dimension independent MCMC sampling methods enchanted by Hessian information.

Finally, randomized and probing algorithms are available to compute the pointwise variance of the prior/posterior distribution and the trace of the covariance operator.

CHAPTER 1

Installation

```
Inverse Problem PYthon library
```



https://hippylib.github.io

`hIPPYlib` depends on FEniCS version 2019.1.

`FEniCS` needs to be built with the following dependecies enabled:

- `numpy`, `scipy`, `matplotlib`, `mpi4py`
- `PETSc` and `petsc4py` (version 3.10.0 or above)
- `SLEPc` and `slepc4py` (version 3.10.0 or above)
- PETSc dependencies: `parmetis`, `scotch`, `suitesparse`, `superlu_dist`, `ml`, `hypre`
- (optional): `mshr`, `jupyter`

1.1 Install hIPPYlib using pip

1.1.1 Latest release

With the supported version of FEniCS and its dependencies installed on your machine, hIPPYlib can be installed via pip as follows

```
pip install hippylib --user
```

In order for pip to install extra requirements (e.g. Jupyter) the following command should be used

```
pip install hippylib[notebook] --user
```

1.1.2 Development version/topic branches

To pip install the development version of hIPPYlib use the command

```
pip install -e git+https://github.com/hippylib/hippylib.git@master#egg=hippylib
```

To pip install a topic branch (e.g. the 2019.1-dev2 branch) use

```
pip install -e git+https://github.com/hippylib/hippylib.git@2019.1-dev2#egg=hippylib
```

> **NOTE:** hIPPYlib applications and tutorials can also be executed directly from the source folder without pip installation.

1.2 Build the hIPPYlib documentation using Sphinx

To build the documentation you need to have sphinx (tested on v.1.7.5), m2r and sphinx_rtd_theme - all of these can be installed via pip.

To build simply run make html from doc folder.

1.3 FEniCS installation

1.3.1 Install FEniCS from Conda (Linux or MacOS)

To use the prebuilt Anaconda Python packages (Linux and Mac only), first install Anaconda3, then run following commands in your terminal:

```
conda create -n fenics-2019.1 -c conda-forge fenics==2019.1.0 matplotlib scipy jupyter
```

> **Note**: FEniCS Anaconda recipes are maintained by the FEniCS community and distributed binary packages do not have a full feature set yet, especially regarding sparse direct solvers and input/output facilities.

1.3.2 Run FEniCS from Docker (Linux, MacOS, Windows)

An easy way to run FEniCS is to use their prebuilt Docker images.

First you will need to install Docker on your system. MacOS and Windows users should preferably use Docker for Mac or Docker for Windows — if it is compatible with their system — instead of the legacy version Docker Toolbox.

Among the many docker's workflow discussed here, we suggest using the Jupyter notebookone.

Docker for Mac, Docker for Windows and Linux users (Setup and first use instructions)

We first create a new Docker container to run the jupyter-notebook command and to expose port 8888. In a command line shell type:

```
docker run --name hippylib-nb -w /home/fenics/hippylib -v $(pwd):/home/fenics/hippylib \
           -d -p 127.0.0.1:8888:8888 hippylib/fenics \
           'jupyter-notebook --ip=0.0.0.0'
docker logs hippylib-nb
```

The notebook will be available at http://localhost:8888/?token=<security_token_for_first_time_connecti in your web browser. From there you can run the interactive notebooks or create a new shell (directly from your browser) to run python scripts.

Docker Toolbox users on Mac/Windows (Setup and first use instructions)

Docker Toolbox is for older Mac and Windows systems that do not meet the requirements of Docker for Mac or Docker for Windows. Docker Toolbox will first create a lightweight linux virtual machine on your system and run docker from the virtual machine. This has implications on the workflow presented above.

We first create a new Docker container to run the jupyter-notebook command and to expose port 8888 on the virtual machine. In a command line shell type:

```
docker run --name hippylib-nb -w /home/fenics/hippylib -v $(pwd):/home/fenics/hippylib \
           -d -p $(docker-machine ip $(docker-machine active)):8888:8888 \
           hippylib/fenics 'jupyter-notebook --ip=0.0.0.0'
docker logs hippylib-nb
```

To find out the IP of the virtual machine we type:

```
docker-machine ip $(docker-machine active)
```

The notebook will be available at http://<ip-of-virtual-machine>:8888/?token=<security_token_for_first_time_connection> in your web browser. From there you can run the interactive notebooks or create a new shell (directly from your browser) to run python scripts.

Subsequent uses

The docker container will continue to run in the background until we stop it:

```
docker stop hippylib-nb
```

To start it again just run:

```
docker start hippylib-nb
```

If you would like to see the log output from the Jupyter notebook server (e.g. if you need the security token) type:

```
docker logs hippylib-nb
```

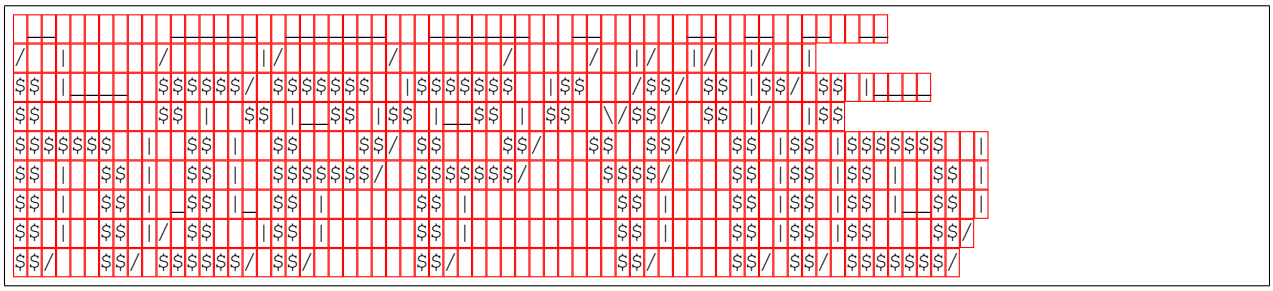
1.3.3 Other ways to build FEniCS

For instructions on other ways to build FEniCS, we refer to the FEniCS project download page. Note that this instructions always refer to the latest version of FEniCS which may or may not be yet supported by hIPPYlib. Always check the hIPPYlib website for supported FEniCS versions.

CHAPTER 2

Changelog

```
Inverse Problem PYthon library
```



https://hippylib.github.io

2.1 Version 3.0.0, released on Feb 2, 2020

- Support for `FEniCS 2019.1.0`
- Modify `PointwiseObservation` so that ordering of observations targets is respected also in parallel. Setting the flag `prune_and_sort` to `True` restores previous behavior.
- Remove unused input `tol` from `Model.solveFwd`, `Model.solveAdj`, `Model.solveFwdIncremental`, `Model.solveAdjIncremental` and from related classes.
- Use `argparse` to set parameters in application drivers from command line
- Use `dl.XDMFFile` to export solutions for visualization in Paraview in all application drivers
- Implement accuracy enhanced SVD algorithm in `randomizedSVD.py`
- Add forward UQ capabilities, using Taylor approximations as control variates
- Introduce hIPPYlib's wrappers to `petcs4py.PETSc.KSP`

- Add reduction operations useful when solving different PDEs concurrently on each process
- Increase coverage of unit testing in CI

2.2 Version 2.3.0, released on Sept 6, 2019

- Reimplement `BiLaplacianPrior` and `MollifiedBiLaplacianPrior` using a more general framework `SqrtPrecisionPDE_Prior`, which also supports Gaussian vector fields.
- Update the prior distribution in `model_subsurf.py` and `tutorial/3_SubsurfaceBayesian.ipynb` to use Robin boundary conditions to alleviate boundary artifacts.
- Update the data misfit term in `model_ad_diff.py` and `tutorial/4_AdvectionDiffusionBayesian.ipynb` to use discrete observations in space and time.

2.3 Version 2.2.1, released on March 28, 2019

- Bug fix missing `mpi_comm` in `TimeDependentVector`
- Bug fix in the initialization of the global variable `parRandom`

2.4 Version 2.2.0, released on Dec 12, 2018

- Add new class `GaussianRealPrior` that implements a finite-dimensional Gaussian prior
- Add a `callback` user-defined function that can be called at the end of each inexact Newton CG iteration
- Add a `__version__` and `version_info` attribute to `hIPPYlib`
- Add `setup.py` to (optionally) install `hIPPYlib` via `pip`
- Add deprecation mechanism
- Deprecate `TimeDependentVector.copy(other)` in favor of `TimeDependentVector.copy()` for consistency with `dolfin.Vector.copy`
- Deprecate `_BilaplacianR.inner(x,y)` for consistency with `dolfin.Matrix`
- CI enhancement via build matrix

2.5 Version 2.1.1, released on Oct 23, 2018

- Update `README.md` and `paper` according to JOSS reviewers's comments
- Add contributing guidelines
- Fix some typos in notebooks (thanks to **Christian Boehm**)

2.6 Version 2.1.0, released on July 18, 2018

- Alleviate boundary artifacts (inflation of marginal variance) in Bilaplacian-like priors using Robin boundary conditions
- Allow the user to select different matplotlib colormaps in jupyter notebooks
- Buxfix in the acceptance ratio of the gpCN MCMC proposal

2.7 Version 2.0.0, released on June 15, 2018

- Introduce capabilities for non-Gaussian Bayesian inference using Mark Chain Monte Carlo methods. Kernels: mMALA, pCN, gpCN, IS. **Note: API subject to change**
- Support domain-decomposition parallelization (new parallel random number generator, and new randomized eigensolvers)
- The parameter, usually labeled `a`, throughout the library, has been renamed to `m`, for `model parameter`. Interface changes:
  - `PDEProblem.eval_da` –> `PDEProblem.evalGradientParameter`
  - `Model.applyWua` –> `Model.applyWum`
  - `Model.applyWau` –> `Model.applyWmu`
  - `Model.applyRaa` –> `Model.applyWmm`
  - `gda_tolerance` –> `gdm_tolerance` in the parameter list for Newton and QuasiNewton optimizers
  - `gn_approx` –> `gass_newton_approx` as parameter in function to compute Hessian/linearization point in classes `Model`, `PDEProblem`, `Misfit`, `Qoi`, `ReducedQoi`
- Organize `hippylib` in subpackages
- Add `sphinx` documentation (thanks to **E. Khattatov** and **I. Ambartsumyan**)

2.8 Version 1.6.0, released on May 16, 2018

- **Bugfix** in `PDEVariationalProblem.solveIncremental` for non self-adjoint models
- Add new estimator for the trace and diagonal of the prior covariance using randomized eigendecomposition
- In all examples and tutorial, use enviromental variable `HIPPYLIB_BASE_DIR` (if defined) to add `hIPPYlib` to `PYTHONPATH`

2.9 Version 1.5.0, released on Jan 24, 2018

- Add support for `FEniCS 2017.2`

2.10 Version 1.4.0, released on Nov 8, 2017

- Add support for `Python 3`
- Enchantments in `PDEVariationalProblem`: it now supports multiple Dirichlet condition and vectorial/mixed function spaces
- Bugfix: Set the correct number of global rows, when targets points fall outside the computational domain
- More extensive testing with `Travis` Integration

2.11 Version 1.3.0, released on June 28, 2017

- Improve `hashdist` installation support
- Switch license to GPL-2
- Add support for `FEniCS 2017.1`

2.12 Version 1.2.0, released on April 24, 2017

- Update instruction to build `FEniCS`: `hashdist` and `docker`
- Update notebook to nbformat 4
- Let `FEniCS 2016.2` be the preferred version of `FEniCS`
- Add `Travis` integration

2.13 Version 1.1.0, released on Nov 28, 2016

- Add partial support for `FEniCS 2016.1` (Applications and Tutorial)
- Improve performance of the randomized eigensolvers

2.14 Version 1.0.2, released on Sep 30, 2016

- Use `vector2Function` to safely convert `dolfin.Vector` to `dolfin.Function`
- Optimize the `PDEVariationalProblem` to exploit the case when the forward problem is linear
- Update notebook `1_FEniCS101.ipynb`

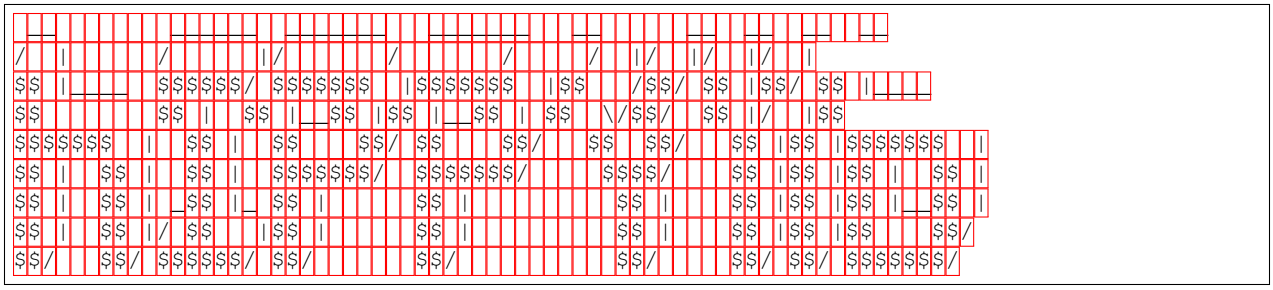
2.15 Version 1.0.1, released on Aug 25, 2016

- Add support in `hippylib.Model` and `hippylib.Misfit` for misfit functional with explicit dependence on the parameter

2.16 Version 1.0.0, released on Aug 8, 2016

- Uploaded to https://hippylib.github.io.
- Initial release under GPL-3.

```
Inverse Problem PYthon library
```



```
https://hippylib.github.io
```

CHAPTER 3

How to Contribute

The `hIPPYlib` team welcomes contributions at all levels: bugfixes, code improvements, new capabilities, improved documentation, or new examples/tutorials.

Use a pull request (PR) toward the `hippylib:master` branch to propose your contribution. If you are planning significant code changes, or have any questions, you should also open an issue before issuing a PR.

See the *Quick Summary* section for the main highlights of our GitHub workflow. For more details, consult the following sections and refer back to them before issuing pull requests:

- *GitHub Workflow*
  - *hIPPYlib Organization*
  - *New Feature Development*
  - *Developer Guidelines*
  - *Pull Requests*
  - *Pull Request Checklist*
- *Automated Testing*
- *Contact Information*

Contributing to hIPPYlib requires knowledge of Git and, likely, inverse problems. If you are new to Git, see the GitHub learning resources. To learn more about inverse problems, see our tutorial page.

By submitting a pull request, you are affirming the Developer's Certificate of Origin at the end of this file.

3.1 Quick Summary

- We encourage you to *join the hIPPYlib organization* and create development branches off `hippylib:master`.
- Please follow the *developer guidelines*, in particular with regards to documentation and code styling.

- Pull requests should be issued toward `hippylib:master`. Make sure to check the items off the *Pull Request Checklist*.
- After approval, hIPPYlib developers merge the PR in `hippylib:master`.
- Don't hesitate to *contact us* if you have any questions.

3.2 GitHub Workflow

The GitHub organization, https://github.com/hippylib, is the main developer hub for the hIPPYlib project.

If you plan to make contributions or will like to stay up-to-date with changes in the code, *we strongly encourage you to *join the hIPPYlib organization**.

This will simplify the workflow (by providing you additional permissions), and will allow us to reach you directly with project announcements.

3.2.1 hIPPYlib Organization

- Before you can start, you need a GitHub account, here are a few suggestions:
  - Create the account at: github.com/join.
  - For easy identification, please add your name and maybe a picture of you at: https://github.com/settings/profile.
  - To receive notification, set a primary email at: https://github.com/settings/emails.
  - For password-less pull/push over SSH, add your SSH keys at: https://github.com/settings/keys.
- *Contact us* for an invitation to join the hIPPYlib GitHub organization.
- You should receive an invitation email, which you can directly accept. Alternatively, *after logging into GitHub*, you can accept the invitation at the top of https://github.com/hippylib.
- Consider making your membership public by going to https://github.com/orgs/hippylib/people and clicking on the organization visibility dropbox next to your name.
- Project discussions and announcements will be posted at https://github.com/orgs/hippylib/teams/everyone.
- The hIPPYlib source code is in the hippylib repository.
- The website is in the web repository.

3.2.2 New Feature Development

- A new feature should be important enough that at least one person, the proposer, is willing to work on it and be its champion.
- The proposer creates a branch for the new feature (with suffix `-dev`), off the `master` branch, or another existing feature branch, for example:

```
# Clone assuming you have setup your ssh keys on GitHub:
git clone git@github.com:hippylib/hippylib.git

# Alternatively, clone using the "https" protocol:
git clone https://github.com/hippylib/hippylib.git
```

(continues on next page)

(continued from previous page)

```
# Create a new feature branch starting from "master":
git checkout master
git pull
git checkout -b feature-dev

# Work on "feature-dev", add local commits
# ...

# (One time only) push the branch to github and setup your local
# branch to track the github branch (for "git pull"):
git push -u origin feature-dev
```

- **We prefer that you create the new feature branch as a fork.** To allow hIPPYlib developers to edit the PR, please enable upstream edits.
- The typical feature branch name is `new-feature-dev`, e.g. `optimal_exp_design-dev`. While not frequent in hIPPYlib, other suffixes are possible, e.g. `-fix`, `-doc`, etc.

3.2.3 Developer Guidelines

- *Keep the code lean and as simple as possible*
  - Well-designed simple code is frequently more general and powerful.
  - Lean code base is easier to understand by new collaborators.
  - New features should be added only if they are necessary or generally useful.
  - Code must be compatible with Python 3.
  - When adding new features add an example in the `application` folder and/or a new notebook in the `tutorial` folder.
  - The preferred way to export solutions for visualization in paraview is using `dl.XDMFFile`
- *Keep the code general and reasonably efficient*
  - Main goal is fast prototyping for research.
  - When in doubt, generality wins over efficiency.
  - Respect the needs of different users (current and/or future).
- *Keep things separate and logically organized*
  - General usage features go in hIPPYlib (implemented in as much generality as possible), non-general features go into external apps/projects.
  - Inside hIPPYlib, compartmentalize between modeling, algorithms, utils, etc.
  - Contributions that are project-specific or have external dependencies are allowed (if they are of broader interest), but should be `#ifdef`-ed and not change the code by default.
- Code specifics
  - All significant new classes, methods and functions have sphinx-style documentation in source comments.
  - Code styling should resemble existing code.
  - When manually resolving conflicts during a merge, make sure to mention the conflicted files in the commit message.

3.2.4 Pull Requests

- When your branch is ready for other developers to review / comment on the code, create a pull request towards `hippylib:master`.
- Pull request typically have titles like:

  ```
  Description [new-feature-dev]
  ```

  for example:

  ```
  Bayesian Optimal Design of Experiments [oed-dev]
  ```

  Note the branch name suffix (in square brackets).
- Titles may contain a prefix in square brackets to emphasize the type of PR. Common choices are: `[DON'T MERGE]`, `[WIP]` and `[DISCUSS]`, for example:

  ```
  [DISCUSS] Bayesian Optimal Design of Experiments [oed-dev]
  ```
- Add a description, appropriate labels and assign yourself to the PR. The hIPPYlib team will add reviewers as appropriate.
- List outstanding TODO items in the description.
- Track the Travis CI *continuous integration* builds at the end of the PR. These should run clean, so address any errors as soon as possible.

3.2.5 Pull Request Checklist

Before a PR can be merged, it should satisfy the following:

- [ ] CI runs without errors.
- [ ] Update `CHANGELOG`:
  - [ ] Is this a new feature users need to be aware of? New or updated application or tutorial?
  - [ ] Does it make sense to create a new section in the `CHANGELOG` to group with other related features?
- [ ] New examples/applications/tutorials:
  - [ ] All new examples/applications/tutorials run as expected.
  - [ ] Add a *fast version* of the example/application/tutorial to Travis CI
- [ ] New capability:
  - [ ] All significant new classes, methods and functions have sphinx-style documentation in source comments.
  - [ ] Add new examples/applications/tutorials to highlight the new capability.
  - [ ] For new classes, functions, or modules, edit the corresponding `.rst` file in the `doc` folder.
  - [ ] If this is a major new feature, consider mentioning in the short summary inside `README` *(rare)*.
  - [ ] If this is a `C++` extension, the `package_data` dictionary in `setup.py` should include new files.

3.3 Automated Testing

We use Travis CI to drive the default tests on the `master` and `feature` branches. See the `.travis` file and the logs at https://travis-ci.org/hi/hippylib.

Testing using Travis CI should be kept lightweight, as there is a 50 minute time constraint on jobs.

- Tests on the `master` branch are triggered whenever a push is issued on this branch.

3.4 Contact Information

- Contact the hIPPYlib team by posting to the GitHub issue tracker. Please perform a search to make sure your question has not been answered already.

3.5 Developer's Certificate of Origin 1.1

By making a contribution to this project, I certify that:

(a) The contribution was created in whole or in part by me and I have the right to submit it under the open source license indicated in the file; or

(b) The contribution is based upon previous work that, to the best of my knowledge, is covered under an appropriate open source license and I have the right under that license to submit that work with modifications, whether created in whole or in part by me, under the same open source license (unless I am permitted to submit under a different license), as indicated in the file; or

(c) The contribution was provided directly to me by some other person who certified (a), (b) or (c) and I have not modified it.

(d) I understand and agree that this project and the contribution are public and that a record of the contribution (including all personal information I submit with it, including my sign-off) is maintained indefinitely and may be redistributed consistent with this project or the open source license(s) involved.

Acknowledgement: We thank the MFEM team for allowing us to use their contributing guidelines file as template.

CHAPTER 4

hippylib

4.1 hippylib package

4.1.1 Subpackages

hippylib.modeling package

Submodules

hippylib.modeling.PDEProblem module

hippylib.modeling.expression module

hippylib.modeling.misfit module

hippylib.modeling.model module

hippylib.modeling.modelVerify module

hippylib.modeling.pointwiseObservation module

hippylib.modeling.posterior module

hippylib.modeling.prior module

hippylib.modeling.reducedHessian module

4.1.2 Module contents