ProbNum: Probabilistic Numerics in Python

Jonathan Wenger, Maren Mahsereci

Dagstuhl Seminar 21432
October 25, 2021

The Next Step for Probabilistic Numerics?

Putting theory into practice.

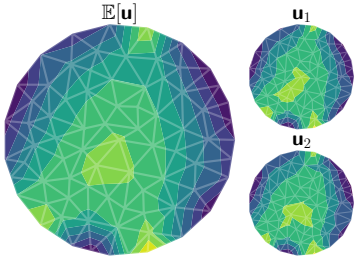
Hennig et al. [1] *"Efficient and stable implementations are still in development. Convincing use-cases from various scientific disciplines are only beginning to emerge."*

Oates and Sullivan [2] *"[…] the coming to fruition of this vision will require demonstrable success on problems that were intractable with the computational resources of previous decades […]."*

Realizing the vision of probabilistic numerics *necessitates* software.

Goals

- demonstrate the *value of a probabilistic approach*
- make use of probabilistic numerical methods *in application*
- propagate uncertainty through *chains of computations*



Software Matters to Research

Software packages advance progress in a research field.

LAPACK · PETSc · NumPy · TensorFlow · PyTorch

Software is a central part of modern science.

Key Advantages

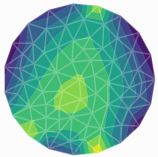
- Demonstration and prototyping
- Reproducible research
- Enables new applications
- Faster method development
- Rapid benchmarking
- Reveals research questions

FEniCS project · PYRO · GPflow · BoTorch

ProbNum

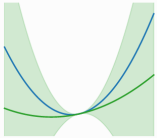
Learn to Approximate. Approximate to Learn.

Get Started 🚀



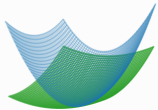
Solve Numerical Problems

Solve problems from linear algebra, optimization, quadrature and differential equations using probabilistic inference.



Quantify Uncertainty in Computation

Quantify and propagate uncertainty from finite resources and stochastic input in computational pipelines.



Compose Custom Numerical Methods

Create problem-specific probabilistic numerical methods from predefined or your own custom components.

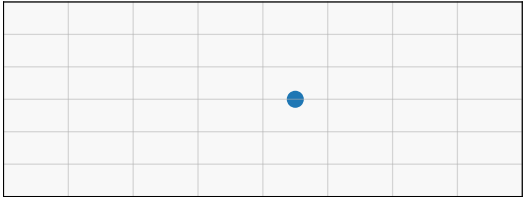
Solving a Linear System

Basic Interface of ProbNum's PN methods.

$$Ax = b$$

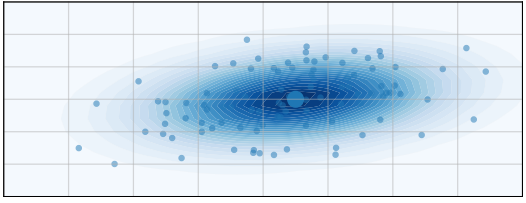
NumPy

```
import numpy as np

x = np.linalg.solve(A, b)
x
```



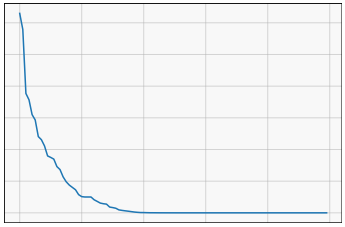
ProbNum

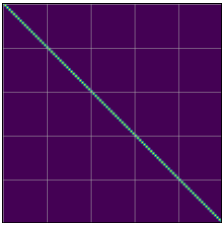
```
import probnum as pn

belief = pn.linalg.problinsolve(A, b)
belief.x
```

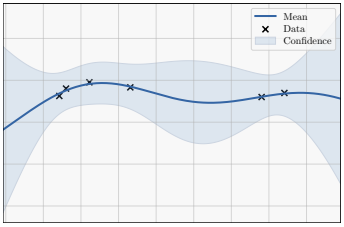


Specifying Prior Information

Encoding prior information from various sources into an algorithm.







`belief.from_spectrum(lmbda)`

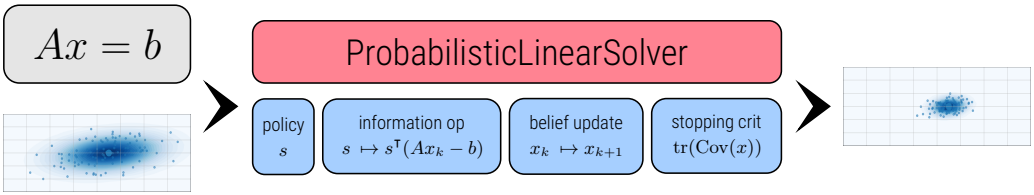
`belief.from_precond(P)`

`belief.from_observations(X)`

Belief constructors define common ways of including prior information.

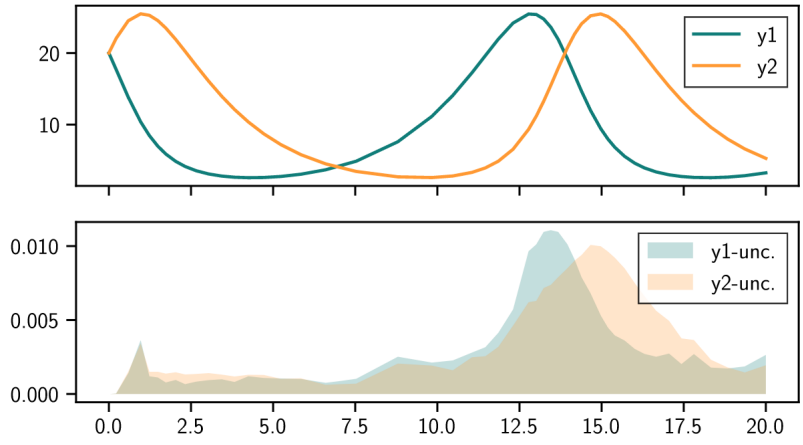
Composing Custom Methods.

Creating new from old.




```python
# Construct custom solver
pls = pn.linalg.ProbabilisticLinearSolver(
    policy=ConjugateGradientPolicy(),
    information_op=ProjectedRHSInformationOp(),
    belief_update=LinearGaussianInferenceBeliefUpdate(),
    stopping_criterion=PosteriorContractionStopCrit(tol=1e-5),
)

# Solve problem
pls.solve(problem=(A, b))
```

Problem: Lotka-Volterra equations (first-order, non-linear ODE)

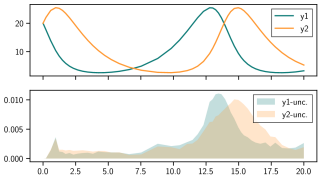


What's in it for me?

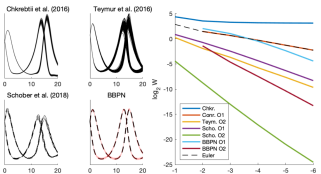
A community software framework benefits *everyone*.

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

demo

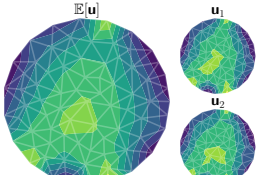


experiment



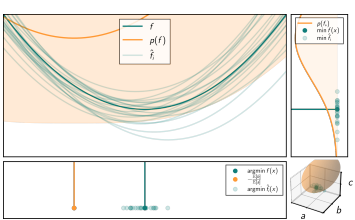
Source: Teymur et al. [5]

apply



Source: Wenger and Hennig [6]

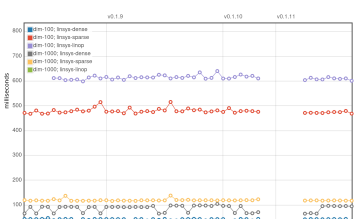
research & develop



implement

```
class ProbabilisticNumericalMethod(ABC, Generic[ProblemType, BeliefType]):
    """Probabilistic numerical methods.

    An abstract base class defining the implementation of a probabilistic numerical
    method [1]_ [2]_. A PN method solves a numerical problem by treating it as a
    probabilistic inference task.

    Parameters
    ----------
    prior :
        Prior knowledge about quantities of interest for the numerical problem.

    References
    ----------
    .. [1] Hennig, P., Osborne, Mike A. and Girolami M., Probabilistic numerics and
        uncertainty in computations. *Proceedings of the Royal Society of London A:
        Mathematical, Physical and Engineering Sciences*, 471(2179), 2015.
    .. [2] Cockayne, J., Oates, C., Sullivan Tim J. and Girolami M., Bayesian
        probabilistic numerical methods. *SIAM Review*, 61(4):756-789, 2019
```

test & benchmark



Join the Effort!

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

ProbNum

- *Solve* numerical problems.
- *Quantify* uncertainty in computation.
- *Compose* custom probabilistic numerical methods.

PyPI		`pip install probnum`
GitHub		`https://github.com/probabilistic-numerics/probnum`
Documentation		`http://probnum.org`
Tutorials		`http://probnum.org/en/latest/tutorials.html`

Questions or Feedback?

References I

[1] Philipp Hennig, Mike A. Osborne, and Mark Girolami. Probabilistic numerics and uncertainty in computations. Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences, 471(2179), 2015.

[2] Chris Oates and TJ Sullivan. A modern retrospective on probabilistic numerics. Statistics and Computing, 10 2019.

[3] Michael Schober, Simo Särkkä, and Philipp Hennig. A probabilistic model for the numerical solution of initial value problems. Statistics and Computing, 29(1):99–122, 2019.

[4] Nathanael Bosch, Philipp Hennig, and Filip Tronarp. Calibrated adaptive probabilistic ODE solvers. In Proceedings of The 24th International Conference on Artificial Intelligence and Statistics (AISTATS), 2021.

[5] Onur Teymur, Christopher N. Foley, Philip G. Breen, Toni Karvonen, and Chris. J. Oates. Black box probabilistic numerics. arXiv preprint, 2021. URL `http://arxiv.org/abs/2106.13718`.

[6] Jonathan Wenger and Philipp Hennig. Probabilistic linear solvers for machine learning. In Advances in Neural Information Processing Systems (NeurIPS), 2020.