ProbNum: Probabilistic Numerics in Python

Maren Mahsereci
maren.mahsereci@uni-tuebingen.de

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Heilbronn ProbNum workshop
London, March 28 2022

Let's talk code

http://probnum.org

- How can users get familiar with PN methods via ProbNum?
- Some examples of functionality. Top-level module overview.
- Topics for this workshop.

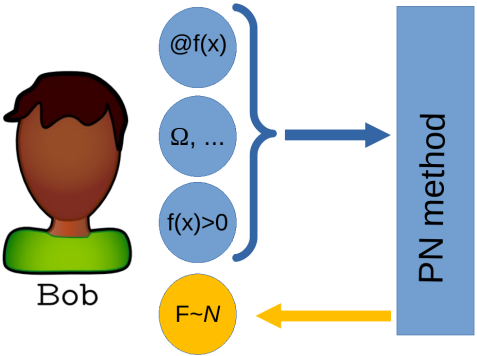
What is a PN method?

Bob wants to infer an integral F. It is given by $F = \int_\Omega f(x)p(x)dx$.

He provides a function handle $@f(x)$ that evaluates the integrand at x when called.
Bob may have further information on the problem, e.g., that $f(x) > 0$ for all $x \in \Omega$.

Bob wants to *use* a probabilistic numerical (PN) method.

What is a PN method?



PN method:

Input:

- Data source: Computational data or data handle related to the quantity of interest.
- Other problem specifications.
- Prior information.

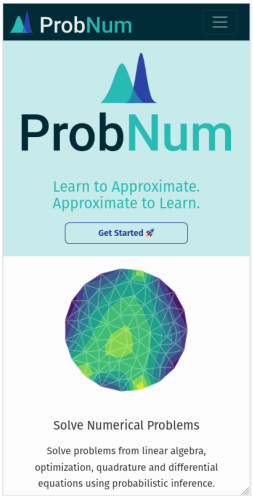
Return:
A random variable object that describes the solution of a non-trivial numerical problem.

Bob wants to infer an integral F. It is given by $F = \int_\Omega f(x)p(x)dx$.

He provides a function handle $@f(x)$ that evaluates the integrand at x when called.
Bob may have further information on the problem, e.g., that $f(x) > 0$ for all $x \in \Omega$.

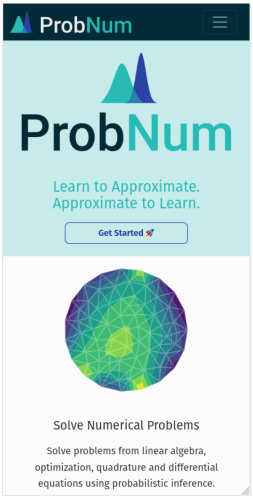
Bob wants to *use* a probabilistic numerical (PN) method.

What is a PN method?

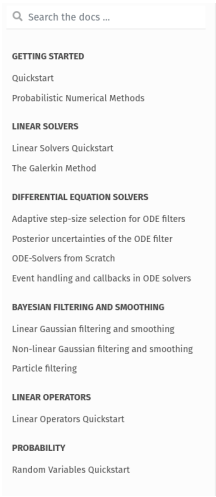


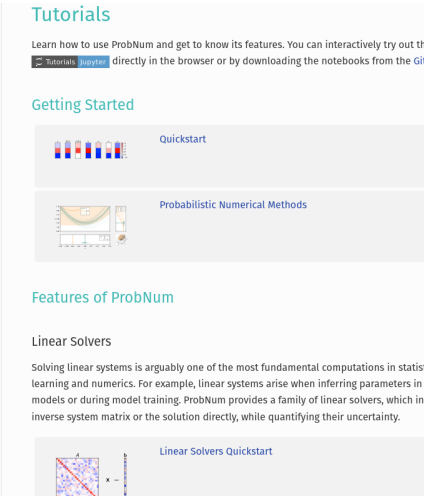
Bob wants to *use* a probabilistic numerical (PN) method.

What is a PN method?



+





Bob wants to *use* a probabilistic numerical (PN) method.

Bob's code 1

From ProbNum's **quickstart** tutorial:

```
from probnum.quad import bayesquad

# define integrand
fun = lambda x: np.sum(x ** 2, axis=1)

# integrate function on domain
F, info = bayesquad(fun=fun, input_dim=1, domain=(0, 1))
```

Bob's code 1

From ProbNum's **quickstart** tutorial:

```
from probnum.quad import bayesquad

# define integrand
fun = lambda x: np.sum(x ** 2, axis=1)

# integrate function on domain
F, info = bayesquad(fun=fun, input_dim=1, domain=(0, 1))
```

Output:

```
>> F: <Normal with shape=(), dtype=float64>
>> F.mean, F.var:  0.3313608243196674 9.98264330309695e-07
>> info:  BQIterInfo(iteration=11, nevals=11, has_converged=True)
```

Bob achieved his goal. **Well documented code attracts users and applications.** [show tutorial]

What is a PN method?

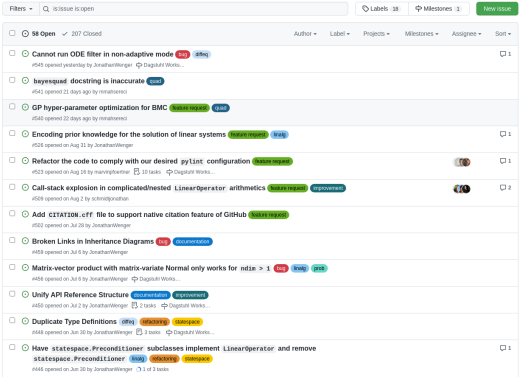
?

Bob wants to infer an integral F. It is given by $F = \int_\Omega f(x)p(x)dx$.

Bob wants to *customize* components of the PN method.

What is a PN method?

?

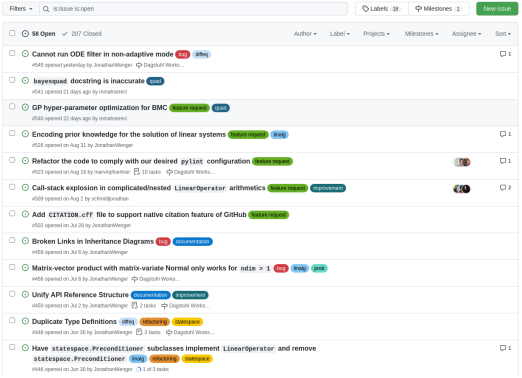


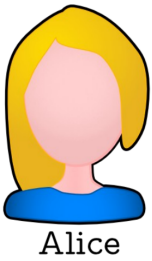
Bob wants to infer an integral F. It is given by $F = \int_\Omega f(x)p(x)dx$.

Bob wants to *customize* components of the PN method. He gets in contact with Alice via GitHub Issues.

What is a PN method?

?

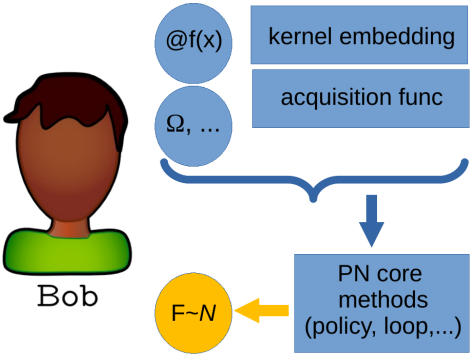




Bob wants to infer an integral F. It is given by $F = \int_\Omega f(x)p(x)dx$.

Bob wants to *customize* components of the PN method. He gets in contact with Alice via GitHub Issues.

What is a PN method?



PN method:

Input:

- Components of the PN method.
- Data source: Computational data or data handle related to the quantity of interest.
- Other problem specifications.

Return:
A random variable object that describes the solution of a non-trivial numerical problem.

Bob wants to infer an integral F. It is given by $F = \int_\Omega f(x)p(x)dx$.

Bob wants to *customize* components of the PN method.
Bob uses the code provided by Alice as guide.

Bob's code 2 (customization)

```
from probnum.randprocs.kernels import Matern
from probnum.quad import *

# define components
kernel = Matern(input_shape=(1, ))
measure = LebesgueMeasure(domain=(0, 1), input_dim=1)
policy = RandomPolicy(measure.sample, batch_size=1, rng=np.random.def
stop_crit = MaxNevals(max_nevals=5)

# create BQ object
bq = BayesianQuadrature(kernel, measure, policy,
                        BQStandardBeliefUpdate(), stop_crit)

# integrate function
F, _, info = bq.integrate(fun=fun, nodes=None, fun_evals=None)
```

Bob's code 2 (customization)

```
...
# create BQ object
bq = BayesianQuadrature(kernel, measure, policy,
                        BQStandardBeliefUpdate(), stop_crit)

# integrate function
F, _, info = bq.integrate(fun=fun, nodes=None, fun_evals=None)
```

Output:

```
>> F: <Normal with shape=(), dtype=float64>
>> F.mean, F.var:0.321592126965595 0.000290959605534713
>> info:  BQIterInfo(iteration=5, nevals=5, has_converged=True)
```

Bob's code 2 (customization)

```
...
# create BQ object
bq = BayesianQuadrature(kernel, measure, policy,
                        BQStandardBeliefUpdate(), stop_crit)

# integrate function
F, _, info = bq.integrate(fun=fun, nodes=None, fun_evals=None)
```

Output:

```
>> F: <Normal with shape=(), dtype=float64>
>> F.mean, F.var:0.321592126965595 0.000290959605534713
>> info:  BQIterInfo(iteration=5, nevals=5, has_converged=True)
```

Bob achieved his goal. **Code educates users.** Well designed research code is intuitive and flexible.

Code quality & contributions

By working with the tutorials, Bob is now a ProbNum user.
He has *increased his understanding* of PN methods.

Code quality & contributions

By working with the tutorials, Bob is now a ProbNum user.
He has *increased his understanding* of PN methods.

During this process, he *found a small bug* in the code:
(Some output in `BQIterInfo` is inconsistent.)

He opens another GitHub Issue and describes the bug and the expected functionality.

Alice confirms the bug and kindly asks Bob to submit a pull request (PR) on GitHub with the corrected code.

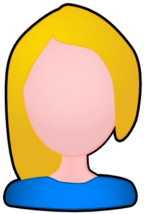
Code quality & contributions

By working with the tutorials, Bob is now a ProbNum user.
He has *increased his understanding* of PN methods.

During this process, he *found a small bug* in the code:
(Some output in `BQIterInfo` is inconsistent.)

He opens another GitHub Issue and describes the bug and the expected functionality.

Alice confirms the bug and kindly asks Bob to submit a pull request (PR) on GitHub with the corrected code.

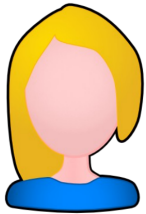


Code quality & contributions

Bob reads ProbNum's *development guide* and creates a pull request (PR) with the code fix.

Alice reviews the code, requests changes, and later approves the code.
The changes are now part of ProbNum's `main` branch.

Bob has improved the quality of ProbNum's code base.
He also is now an official contributor of ProbNum and has augmented his portfolio and CV.

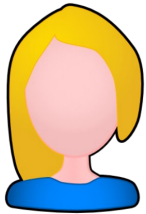


Code quality & contributions

Bob reads ProbNum's *development guide* and creates a pull request (PR) with the code fix.

Alice reviews the code, requests changes, and later approves the code.
The changes are now part of ProbNum's `main` branch.

Bob has improved the quality of ProbNum's code base.
He also is now an official contributor of ProbNum and has augmented his portfolio and CV.



Users augment functionality and **increase robustness and quality** of a code base. [show dev/PR]

GitHub enables contributions via Issues and pull requests (PRs) under controlled procedures.

GitHub Actions enable **Continuous Integration** (CI) via automated tests and code-format checks.
These ensure **high code standards** which in return increase **user trust**.

ProbNum uses `tox` to unify the local development with CI builds.

Summary of Bob's interaction with ProbNum

- Users have a variety of goals. ProbNum has APIs for different user experiences (e.g., from-problem-description vs. custom vs. dev).
- Well maintained code attracts users and increases general understanding of PN methods.
- Users contribute: They augment functionality and increase code robustness and quality. This, in return, increases user trust.

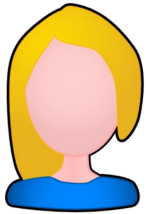
GitHub facilitates all processes.

What's in for me?

Benefits of open source libraries in modern research

Open source software is an integral part of modern research.

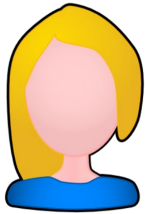
- BLAS, LAPACK, Python, NumPy, SciPy, ...
- PyTorch, TensorFlow, JAX, Theano, Keras,
- Stan, Pyro, TensorFlow Probability, ...
- GPy, GPyTorch, GPflow, ...
- GPyOpt, BOtorch, EmuKit, ... **ProbNum**



What's in for me?

Benefits of open source libraries in modern research

- Showcase & demonstrate.
- Apply immediately.
- Compare, benchmarks, reproduce, experiment.
- Prototype, develop fast, build on existing components.
- Make you research accessible, reusable.
- Deliver quality code (unittests, reviews, CI), increase trust.
- Discover new research questions.



Alice

- Use for teaching & education.
- Visibility and trust of field. Realize vision.
- Share maintenance, use synergies.
- For individuals: Invest, independent of position, use in grant proposals. Students: learn skill, gain experience, augment CV.

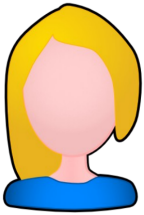
What's in for me?

Benefits of open source libraries in modern research

Eve

- Showcase & demonstrate.
- Apply immediately.
- Compare, benchmarks, reproduce, experiment.
- Prototype, develop fast, build on existing components.
- Make you research accessible, reusable.
- Deliver quality code (unittests, reviews, CI), increase trust.
- Discover new research questions.

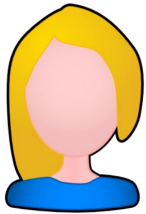
- Use for teaching & education.
- Visibility and trust of field. Realize vision.
- Share maintenance, use synergies.
- For individuals: Invest, independent of position, use in grant proposals. Students: learn skill, gain experience, augment CV.



Alice

More than just solvers

Top-level modules of ProbNum



More than just solvers

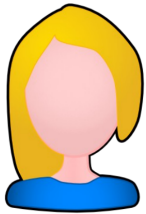
Top-level modules of ProbNum

PN solvers

- `pn.diffeq`
  - `pn.diffeq.probsolve_ivp`
  - `pn.diffeq.perturbsolve_ivp`
  - ...
- `pn.quad`
  - `pn.quad.bayesquad`
  - `pn.quad.bayesquad_from_data`
  - ...
- `pn.linalg`
  - `pn.linalg.problinsolve`
  - `pn.linalg.bayescg`
  - ...

Supporting packages

- `pn.randvars`
  - `pn.randvars.Normal`
  - ...
- `pn.randprocs`
  - `pn.randprocs.kernels`
  - `pn.randprocs.GaussianProcess`
  - ...
- `pn.linops`
  - `pn.linops.Kronecker`
  - ...
- `pn.filtsmooth`
  - `pn.filtsmooth.filter_kalman`
  - ...



[sow API refs]

More than just solvers

Top-level modules of ProbNum

Supporting modules provide functionality used by the PN solvers.

Modularity has benefits:

- Re-purpose: Modules are general enough to be of use elsewhere.
  - Linear operators & randvars
  - Random processes & kernels
  - Filters and smoothers
  - ...
- Separation of concerns is intuitive.
- Building on existing, well-tested components is a good idea and saves time.

Supporting packages

`pn.randvars`

`pn.randvars.Normal`

...

`pn.randprocs`

`pn.randprocs.kernels`

`pn.randprocs.GaussianProcess`

...

`pn.linops`

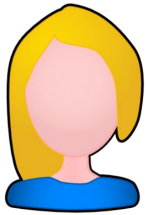
`pn.linops.Kronecker`

...

`pn.filtsmooth`

`pn.filtsmooth.filter_kalman`

...



Alice

Bob

[sow API refs]

Examples

Example 1: Create & transform random variables

```
from probnum.randvars import Normal

# define random variable
x_rv = Normal(mean=0., cov=1.)

# affine transformation
y_rv = 2 * x_rv + 1
```

Output:

```
>> y_rv:  <Normal with shape=(), dtype=float64>
>> F.mean, F.var:  1.0 4.0
```

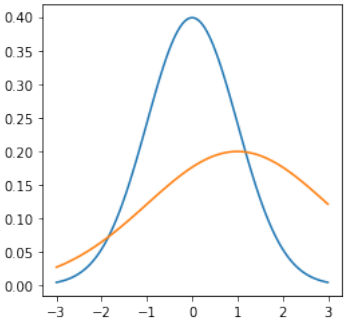
Example 1: Create & transform random variables

```
from probnum.randvars import Normal

# define random variable
x_rv = Normal(mean=0., cov=1.)

# affine transformation
y_rv = 2 * x_rv + 1
```

Output:

```
>> y_rv:  <Normal with shape=(), dtype=float64>
>> F.mean, F.var:  1.0 4.0
```

ProbNum provides random variable arithmetics.



Example 2: Create & transform random variables (multi-dim)

```
from probnum.linops import Matrix

#define random variable
x_rv = Normal(mean=np.array([1., 2., 3.]),
    cov=np.diag(np.array([4., 5., 6.])))

# define linear operators from matrix
P = np.array([[1, 0, 0], [0, 1, 0]])
Pop = Matrix(P)

# transform
y_rv = Pop @ x_rv
```

Output:

```
>> y_rv:  <Normal with shape=(2,), dtype=float64>
>> y_rv.mean, y_rv.var:  [1., 2.]  [4.0, 5.0 ]
```

Example 2: Create & transform random variables (multi-dim)

```
from probnum.linops import Matrix

#define random variable
x_rv = Normal(mean=np.array([1., 2., 3.]),
    cov=np.diag(np.array([4., 5., 6.])))

# define linear operators from matrix
P = np.array([[1, 0, 0], [0, 1, 0]])
Pop = Matrix(P)

# transform
y_rv = Pop @ x_rv
```

Output:

```
>> y_rv:  <Normal with shape=(2,), dtype=float64>
>> y_rv.mean, y_rv.var:  [1., 2.]  [4.0, 5.0 ]
```

RVs can be transformed by applying scalars, np.ndarrays or instances of `LinearOperator` using overloaded arithmetic operators (`*`, `+`, `@`, `..`).

This enables easy to read, but efficient RV manipulation.

Example 3: Matrix-free linear operators

```
# define matrix-vector product
@LinearOperator.broadcast_matvec
def mv(v):
    return np.roll(v, 1)  # shifts by one

# create linear operator from mv
Aop = LinearOperator(shape=(5, 5),
            dtype=np.float_, matmul=mv)

# apply to vector (or RV)
x = np.arange(0., 5, 1)
y = Aop @ x
```

Output:

```
>> x:  [0., 1., 2., 3., 4.]
>> y:  [4., 0., 1., 2., 3.]
```

Example 3: Matrix-free linear operators

```
# define matrix-vector product
@LinearOperator.broadcast_matvec
def mv(v):
    return np.roll(v, 1)  # shifts by one

# create linear operator from mv
Aop = LinearOperator(shape=(5, 5),
                     dtype=np.float_, matmul=mv)

# apply to vector (or RV)
x = np.arange(0., 5, 1)
y = Aop @ x
```

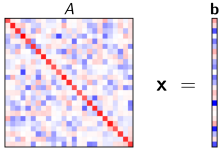
Output:

```
>> x:  [0., 1., 2., 3., 4.]
>> y:  [4., 0., 1., 2., 3.]
```

Often it is sufficient to encode the matrix-vector product of an operator.

This enables *compute- and memory-efficient* custom linear operators.

The dense matrix can still be constructed if required.



Example 4: Sparse linear operators

```
import scipy.sparse

# create a sparse matrix using SciPy
A_scipy = scipy.sparse.rand(m=5, n=5,
        density=0.05, random_state=42)

# create a ProbNum linear operator
Aop = Matrix(A=A_scipy)

# apply to vector (or RV)
x = np.ones(5)
y = Aop @ x
```

Output:

```
>> x:  [1., 1., 1., 1., 1.]
>> y:  [0., 0., 0., 0.30424224, 0.]
```

Example 4: Sparse linear operators

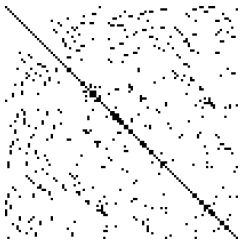
```
import scipy.sparse

# create a sparse matrix using SciPy
A_scipy = scipy.sparse.rand(m=5, n=5,
        density=0.05, random_state=42)

# create a ProbNum linear operator
Aop = Matrix(A=A_scipy)

# apply to vector (or RV)
x = np.ones(5)
y = Aop @ x
```

Output:

```
>> x:  [1., 1., 1., 1., 1.]
>> y:  [0., 0., 0., 0.30424224, 0.]
```

Create linear operators from SciPy's sparse matrices.

Use their efficient implementation.
It carries over to ProbNum.



Example 5: Kronecker product & Matrix-Normal

```
# Define the mean matrix
n = 20  # matrix-variate RV is nxn
mean = np.eye(n)

# Define the Kronecker covariance matrix
V = 1 / k * scipy.sparse.diags(..,
            shape=(n, n)).toarray()
W = np.eye(n)
cov = Kronecker(A=V, B=W)

# create matrix-variate normal RV
X_rv = Normal(mean=mean, cov=cov)
```

Output:

```
>> X_rv:  <Normal with shape=(100, 100)..>
>> X_rv.cov:  <Kronecker with shape=(100, 100)..>
```

Example 5: Kronecker product & Matrix-Normal

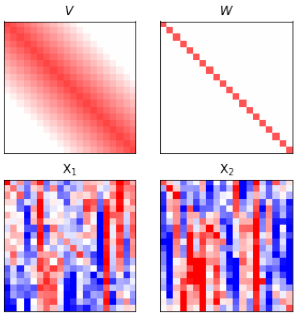
```
# Define the mean matrix
n = 20  # matrix-variate RV is nxn
mean = np.eye(n)

# Define the Kronecker covariance matrix
V = 1 / k * scipy.sparse.diags(..,
              shape=(n, n)).toarray()
W = np.eye(n)
cov = Kronecker(A=V, B=W)

# create matrix-variate normal RV
X_rv = Normal(mean=mean, cov=cov)
```

Output:

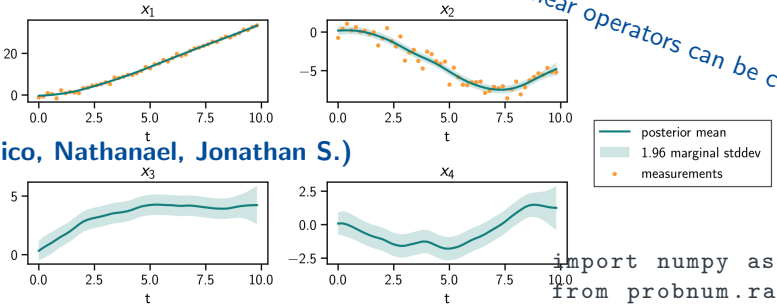
```
>> X_rv:  <Normal with shape=(100, 100)..>
>> X_rv.cov:  <Kronecker with shape=(100, 100)..>
```

`Kronecker` is a ProbNum `LinearOperator`.
It can be used as covariance matrix in `Normal`.

Samples obey the Kronecker covariance:



More examples: Filtering & smoothing, kernel arithmetics, ...

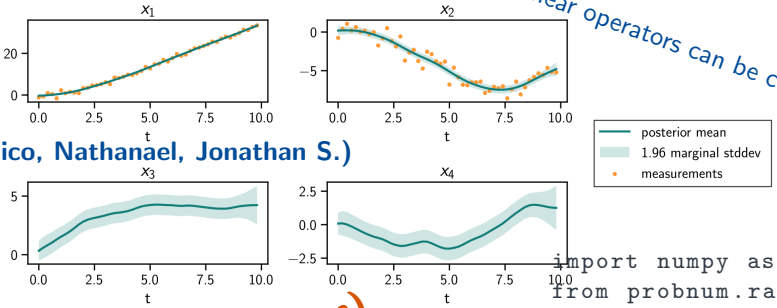


Linear operators can be combined (`A @ B + C`).

(Nico, Nathanael, Jonathan S.)

```
import numpy as np
from probnum.randprocs.kernels \
        import ExpQuad, WhiteNoise

# Kernel via arithmetic
k = ExpQuad(input_shape=()) + \
    0.01 * WhiteNoise(input_shape=())
```

More examples: Filtering & smoothing, kernel arithmetics, ...



(Nico, Nathanael, Jonathan S.)

Linear operators can be combined (`A @ B + C`).

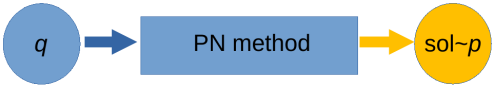
Auto-diff backend (soon)
(Marvin, Jonathan W.)

```
import numpy as np
from probnum.randprocs.kernels \
        import ExpQuad, WhiteNoise

# Kernel via arithmetic
k = ExpQuad(input_shape=()) + \
    0.01 * WhiteNoise(input_shape=())
```

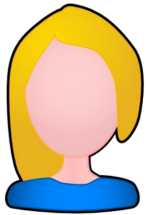
Towards propagating uncertainty

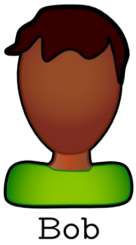
ProbNum aims to not only develop solver modules, but a high-level structure.

- The PN-solvers' in- and output objects (`ranvars`, `randprocs`, ...)
- A loose `core` module with abstract components to allow composability (`policy`, `stopping_criterion`, ...).

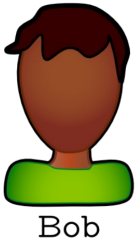


- ProbNum is a greenfield project (there is no similar library yet).
- Great opportunity to learn about the practical aspects of realizing parts of the PN vision.





Developing custom component (example)



Developing custom component (example)

```
# Abstract class for stopping criterion.
class StoppingCriterion(abc.ABC):

    @abc.abstractmethod
    def __call__(self, *args, **kwargs) -> bool:
        raise NotImplementedError

    def __and__(self, other):
        return LambdaStoppingCriterion(
            stopcrit=lambda *args, **kwargs:
            self(*args, **kwargs) and other(*args, **kwargs))

    def __or__(self, other):
        return LambdaStoppingCriterion(
            stopcrit=lambda *args, **kwargs:
            self(*args, **kwargs) or other(*args, **kwargs))

    def __invert__(self): ...
```

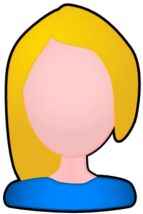
Developing custom component (example)

```
# Stopping criterion specific to a linear solver
class ResidualNormStoppingCriterion(StoppingCriterion):

    def __init__(self, atol = 10**-5, rtol = 10**-5,):
        self.atol = pn.utils.as_numpy_scalar(atol)
        self.rtol = pn.utils.as_numpy_scalar(rtol)

    def __call__(self, solver_state) -> bool:
        res_norm = np.linalg.norm(solver_state.residual,
                                  ord=2)
        b_norm = np.linalg.norm(solver_state.problem.b,
                                ord=2)

        return res_norm <= self.atol or \
               res_norm <= self.rtol * b_norm
```

A detour on common industry hurdles

Applying PN methods in real applications

Eve



Alice

A detour on common industry hurdles

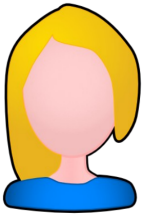
Applying PN methods in real applications

Eve would like to *apply* PN methods.

Initial hurdles:

- Install & get familiar with calling a solver ($<$ 15 mins).
- Understand what the *input* and *output* objects represent. ($<$ 30 mins).
- Later: Set up workflow ($<$ 30 mins).

Eve



Alice

A detour on common industry hurdles

Applying PN methods in real applications

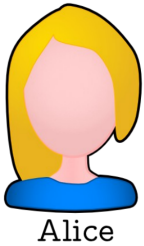
Eve would like to *apply* PN methods.

Initial hurdles:

- Install & get familiar with calling a solver ($<$ 15 mins).
- Understand what the *input* and *output* objects represent. ($<$ 30 mins).
- Later: Set up workflow ($<$ 30 mins).

She needs to be sure that:

- The *code does not fail* most of the time.
- The *results are good* without "tweaking" most of the time.
- There are *theoretical guarantee*.



A detour on common industry hurdles

Applying PN methods in real applications

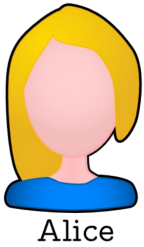
Eve would like to *apply* PN methods.

Initial hurdles:

- Install & get familiar with calling a solver ($<$ 15 mins).
- Understand what the *input* and *output* objects represent. ($<$ 30 mins).
- Later: Set up workflow ($<$ 30 mins).

She needs to be sure that:

- The *code does not fail* most of the time.
- The *results are good* without "tweaking" most of the time.
- There are *theoretical guarantee*.



Industry users often **do not have a lot of time** to explore beyond their project.
If the RoI is not guaranteed, the initial **hurdles must be low**.
Performance (reliably good solver results) and **code robustness are key.** Averages matter. [dev guide]

ProbNum Zoo: Test problems for PN methods

- `pn.problems.zoo` collects test problems for PN solvers.
- Unified API (ready to use with solvers).

- Demonstrate on toy problems.
- Showcase robustness of method by running it on many problems.
- Easy paper writing.

- Might enable benchmarking later.

Aims for this workshop

https://github.com/probabilistic-numerics/probnum

State

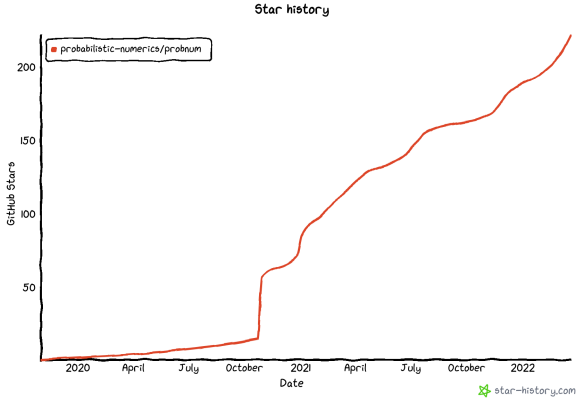
- The package is growing.

Aims for this workshop

https://github.com/probabilistic-numerics/probnum

State

- The package is growing.



Aims for this workshop

https://github.com/probabilistic-numerics/probnum

State

- The package is growing.

Aims

- Increase familiarity with ProbNum in PN community.
  - Interact with the tutorials. Add missing tutorial.
  - Add test functions to `pn.problems.zoo` (example: F-X for `quad`).
  - Use ProbNum as dependency in your project (e.g., `LinearOperator`, ... functionality)

Involvement increasingly "federal".

- Increase support of individual modules (+planning exercise).
  - `pn.diffeq` Nico, Nathanael, Jonathan S. (Marvin)
  - `pn.linalg` Jonathan W., Marvin, Tim R., (Jon?)
  - `pn.quad` Toni, Maren, (Alex?, F-X?, Masha?)

  Anyone interested is welcome. This does not need to be a big commitment.

Aims for this workshop

https://github.com/probabilistic-numerics/probnum

State

- The package is growing.

Aims

- Increase familiarity with ProbNum in PN community.
  - Interact with the tutorials. Add missing tutorial.
  - Add test functions to `pn.problems.zoo` (example: F-X for `quad`).
  - Use ProbNum as dependency in your project (e.g., `LinearOperator`, ... functionality)

Involvement increasingly "federal".

- Increase support of individual modules (+planning exercise).
  - `pn.diffeq` Nico, Nathanael, Jonathan S. (Marvin)
  - `pn.linalg` Jonathan W., Marvin, Tim R., (Jon?)
  - `pn.quad` Toni, Maren, (Alex?, F-X?, Masha?)

  Anyone interested is welcome. This does not need to be a big commitment.

Thank you!

maren.mahsereci@uni-tuebingen.de

http://probnum.org