DTU Compute
Department of Applied Mathematics and Computer Science

DTU

CUQIpy-Benchmarks: A new collection of UQ benchmarks

Naoki Sakai[1], Tania Goia[2], Amal M. A. Algamdi[3], Chao Zhang[3], Jakob S. Jørgensen[3,*]

[1] The University of Tokyo, [2]Technical University of Hamburg, [3] Technical University of Denmark,[*] Corresponding authors

1 Introduction

This project focuses on benchmarking the sampling methods in CUQIpy [2].

A good benchmark is neither too simple, nor too complicated, while also being able to reflect properties of real life applications [1].

Aim

- create a collection of benchmarks for CUQIpy
- sampling with different methods simultaneously
- compare the methods by different criterias
- visualize the distribution (in case of 1D and 2D)

2 Types of benchmarks

1. **Distribution based benchmarks**
   - a specified target distribution
   - not inverse problem

   Donut: the target distribution
   $\log(p(\mathbf{x})) \propto -\frac{1}{\sigma^2}(\|\mathbf{x}\|-r)^2$

   Sixmodal [3]: the target distribution
   $\log(p(x,y)) \propto -\frac{x^2}{2} - \frac{(\csc^5(y)-x)^2}{2}$

2. **Inverse Problem benchmarks**
   - Bayesian inverse problem (BIP)
   - leads to a specified target distribution
   - has likelihood, posterior, prior
   - can be a PDE problem

   World's Simplest BIP:Linear-Gaussian model

   Heat 1D:Heat equation with noisy measurement

3 Process

How it works

INPUTS	ALGORITHM	OUTPUTS
- methods - criteria - parameters	- generates samples - flexible - multi dimensional	- table of values -representative images

Key Features

- **flexibility**: the user can choose the criteria and sampling methods they desire:

```
methods = ["ULA"], criteria=["ESS"]
```

- **simplicity**: the user can easily test the benchmarks via classes:

```
import benchmarksClass as BC
target_donut = BC.Donut()
```

- **versatility**: the sampling methods can be configured in different ways

The Sampling Methods

Non-gradient based

- Random Walk Metropolis Hastings (MH)
- Component-wise Metropolis Hastings (CWMH)

Gradient based

- Unadjusted Langevin algorithm (ULA)
- Metropolis-adjusted Langevin algorithm (MALA)
- No-U-Turn Sampler (NUTS)

Criteria

ESS	Effective Sample Size Higher is better
Acc.Rate	Acceptance Rate Ideal value is dependent on methods
RHAT	Rank normalized split R-hat for multiple chains Closer to 1 is better
LogPDF	How many times logpdf is computed Lower is better
Gradient	How many times gradient is computed Lower is better

4 The CUQIpy-Benchmarks collection

We present two examples from our repository, available at https://github.com/CUQI-DTU/CUQIpy-Benchmarks.

You can also access the repository by scanning the QR code at bottom right of this poster.

1. **D01-donut-table.ipynb**

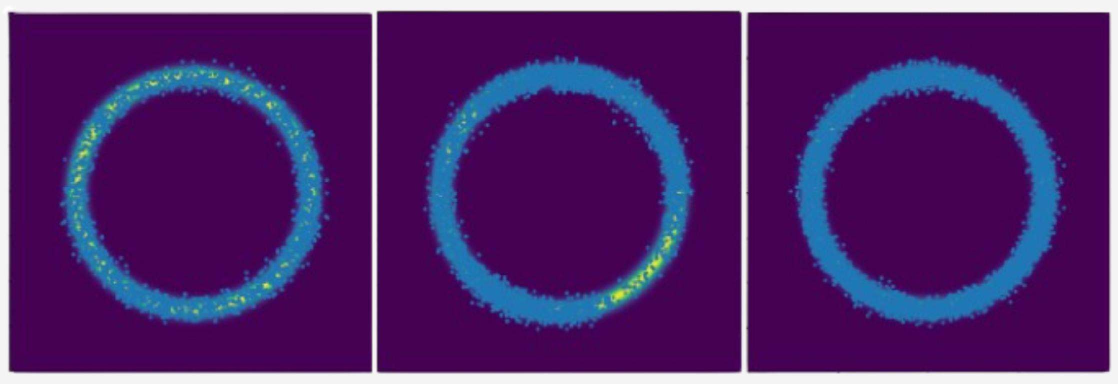

Figure 1: MH, MALA, NUTS sampling

	MH	MALA	NUTS
Samples	8500	8500	8500
Burn-ins	1500	1500	1500
ESS	36.35	10.1	3084.9
Acc. Rate	0.223	0.506	0.784
RHAT	1.035	1.068	1.0
LogPDF/ESS	275.1	990.1	94.19
Gradient/ESS	0	990.1	94.19

Table 1: Comparison for Donut

- MH: accurate, but sparse sampling
- MALA: high computational cost, and somewhat inaccurate results
- NUTS: accurate sample representation

2. **D03-sixmodal.ipynb**

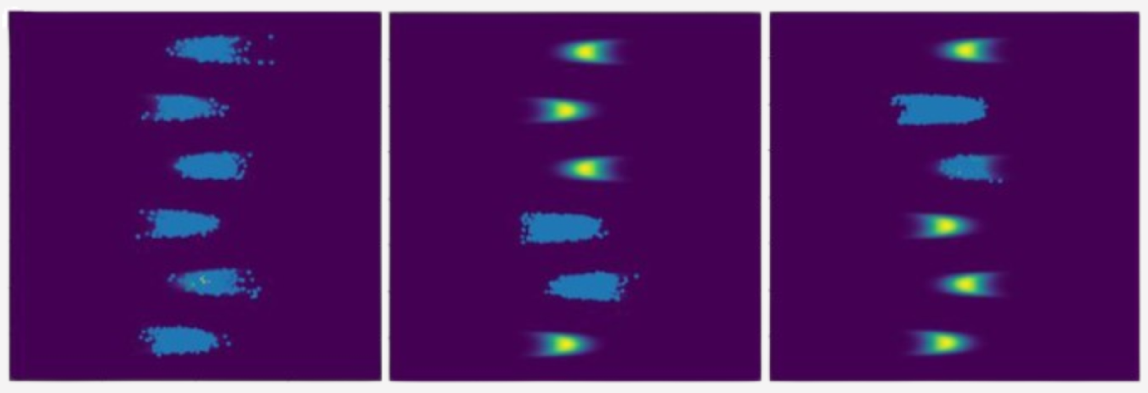

Figure 2: MH, MALA, NUTS sampling

	MH	MALA	NUTS
ESS	39.05	2.2	349.18
Acc. Rate	0.248	0.492	0.987
RHAT	1.075	1.982	1.35
LogPDF/ESS	256.08	4545.45	4842.69
Gradient/ESS	0	4545.45	4842.69

Table 2: Comparison for Sixmodal

- NUTS & MALA : high computational costs
- MH: lower cost & explores every mode
- RHAT high (MALA , NUTS) → lack of exploration of modes

5 Conclusion and Future Work

What we did: developed a comprehensive benchmark collection that simplifies the evaluation of the CUQIpy sampling methods.

Interesting finding: gradient-based methods, which are often perceived as more efficient, may sometimes perform less accurately than non-gradient-based methods.

Looking ahead: enhance our suite by integrating additional problems, including the one proposed in [1].

References

[1] David Aristoff and Wolfgang Bangerth. "A Benchmark for the Bayesian Inversion of Coefficients in Partial Differential Equations". *Computational Methods in Applied Mathematics*, 21(1):23–51, 2021.

[2] Nicolai A. B. Riis et al. "CUQIpy: I. Computational Uncertainty Quantification for Inverse Problems in Python". *Inverse Problems*, 40(4):045009, 2024.

[3] Vivekananda Roy. "Convergence Diagnostics for Markov Chain Monte Carlo". *Annual Review of Statistics and Its Application*, 7:387–412, 2020.

CUQI VILLUM FONDEN CUQIpy