CUQIpy

Computational **U**ncertainty **Q**uantification for **I**nverse problems in **py**thon

Joint work with:
Babak Afkham
Silja L. Christensen
Felipe Uribe
Per Christian Hansen
and CUQI

Jakob Sauer Jørgensen | Nicolai Riis | Amal Alghamdi
Technical University of Denmark (DTU)

SIAM UQ24 | MS71
Computational Tools for Large-Scale Inverse Problems and UQ
Trieste, Italy | 28 February 2024

Who? CUQIpy Funding, Developers and Contributors

CUQI VILLUM FONDEN

CUQI project at DTU (2019-2025): **C**omputational **U**ncertainty **Q**uantification for **I**nverse Problems

CUQI Project PI

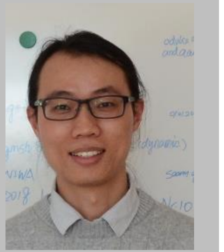
Per Christian Hansen
Professor

CUQIpy core developers

Jakob Jørgensen
Senior Researcher

Nicolai Riis
Postdoc

Amal Alghamdi
Postdoc



Chao Zhang
Postdoc

CUQIpy major contributors

Babak Afkham
Postdoc

Silja Christensen
PhD Student

Felipe Uribe
Former Postdoc

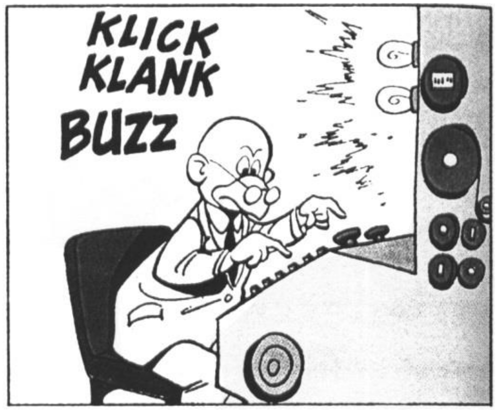
CUQI team in 2021. Team uses software, provides feedback and contributes theory & code.

CUQIpy in a Nutshell

Vision

Build a software package that uses uncertainty quantification (UQ) to access and quantify uncertainties in solutions to **imaging** inverse problems.

- **Simplify** the mathematics, statistics and code for the non-expert user.
- Provide **full control** for expert users.
- Allow users to focus on **modeling aspects**.
- UQ in **five lines of code!**



Features

- Easy access to **state-of-the-art** tools in one framework (including 3rd party libraries).
- A suite of **test problems** to allow users to get started.
- Allow users to provide **custom code** for models, distributions, samplers etc.
- Exploit structure to support **large-scale** problems.

Why not use an existing software package?

General UQ software:

- Tends to break down for large-scale imaging-type problems.

Software for UQ in inverse problems:

- Often specialized for certain types of problems.

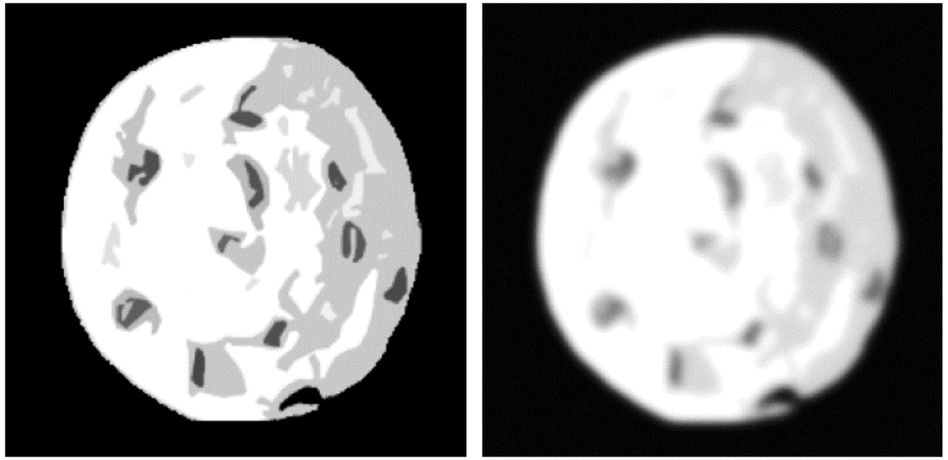
The niche that CUQIpy is aimed at:

- Unified interface for broad range of imaging problems.
- Simple “non-expert” interface.
- Test problem suite.
- Interface to other software libraries.
- Support user-defined code.

Cookie deblurring with CUQIpy

$$Ax = y$$

True | Blurred, noisy



`info.exactSolution.plot()` `y_obs.plot()`

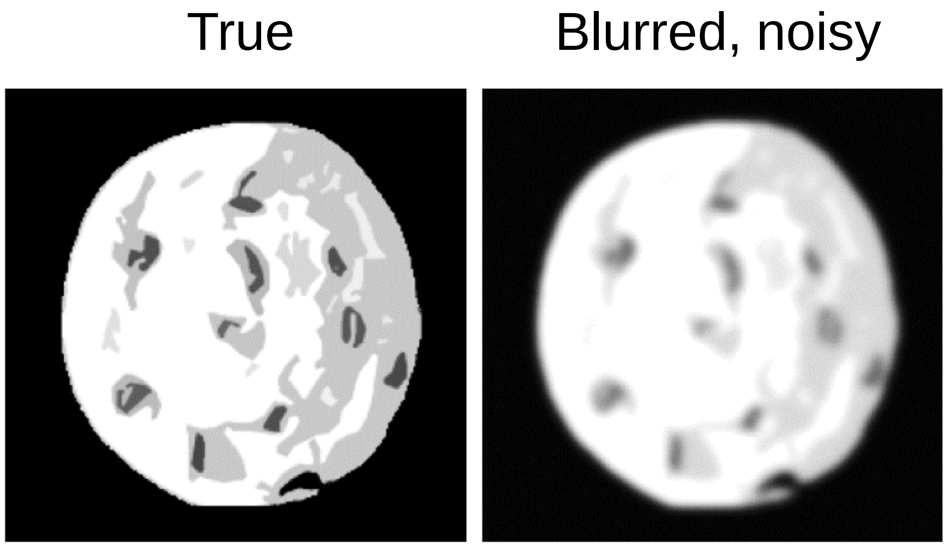
Using testproblem library

```
A, y_obs, info = Deconvolution2D(dim=512, phantom="cookie")
```

```
print(A)
>>> CUQI LinearModel: Image2D(512,512) -> Image2D(512,512). Forward parameters: ['x']
```

Cookie deblurring with CUQIpy

$$Ax = y$$



`info.exactSolution.plot()`

`y_obs.plot()`

Using custom forward model

```
A = LinearModel(forward_func,
                adjoint_func,
                range_geometry=Image2D((512,512)),
                domain_geometry=Image2D((512,512)))
```

User-defined code

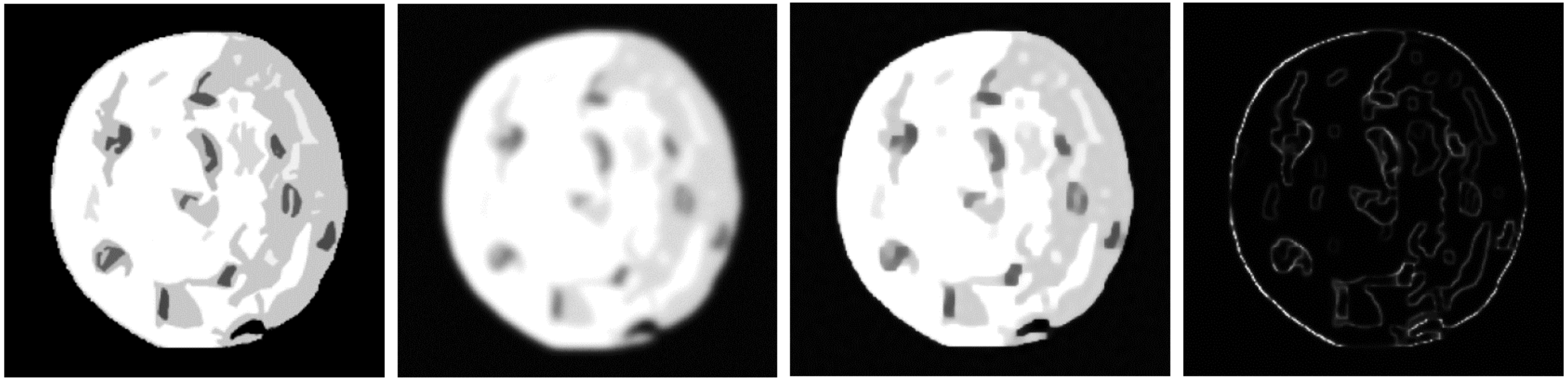
```
print(A)

>>> CUQI LinearModel: Image2D(512,512) -> Image2D(512,512). Forward parameters: ['x']
```

Cookie deblurring with CUQIpy

$$Ax = y$$

True | Blurred, noisy | Posterior mean | Posterior std



$$d \sim \mathrm{Gamma}(1, 10^{-4})$$
$$s \sim \mathrm{Gamma}(1, 10^{-4})$$
$$\boldsymbol{x} \sim \mathrm{LMRF}(d^{-1}),$$
$$\boldsymbol{y} \sim \mathrm{Gaussian}(\boldsymbol{A}\boldsymbol{x}, s^{-1}\boldsymbol{I})$$

Laplace Markov Random Field

```
d = Gamma(1,1e-4)
s = Gamma(1,1e-4)
x = LMRF(1/d, geometry=A.domain_geometry)
y = Gaussian(A @ x, 1/s)
BP = BayesianProblem(x, y, d, s)
BP.set_data(y=y_obs)
BP.UQ()
```

Overview of CUQIpy Modules and Plug-ins

CUQIpy

- Model (forward)
- Distribution
- Random variable
- Sampler
- Geometry
- Likelihood
- Density
- Test problem
- Array
- ....

plug-ins

- CUQIpy-CIL
- CUQIpy-FEniCS
- CUQIpy-PyTorch
- ....

CUQIpy: https://cuqi-dtu.github.io/CUQIpy/

CUQIpy geometry

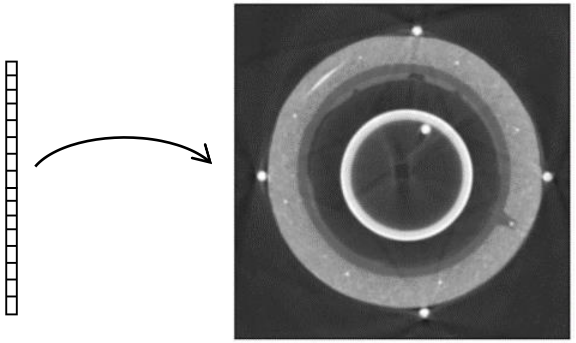
The `Geometry` object represents the spaces of the model domain and range

- ➢ Maps parameters to model input (function values).
- ➢ Contains tools for visualization.
- ➢ Provides an interface to connect to 3rd party libraries.

`x.plot()`

`Image2D:` $\mathbf{x} \mapsto \mathbf{X}$

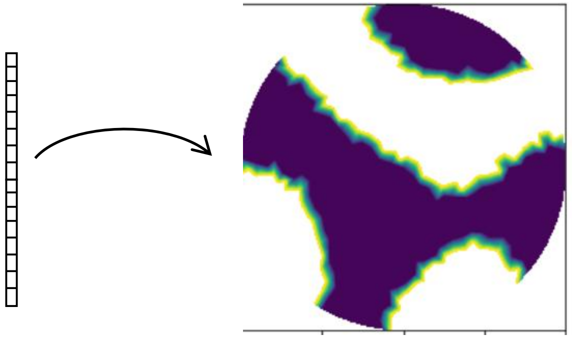
Parameters (vector)



Function values (2D image, matrix)

`FEniCSKL:` $\boldsymbol{\theta} \mapsto h\left(\sum_{i\in\mathbb{N}} \sqrt{\lambda_i}\theta_i e_i(x)\right)$

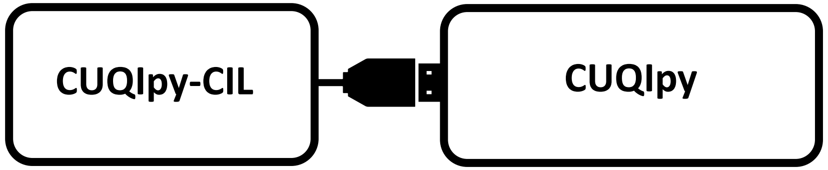
Parameters (vector)



Function values (FEniCS mesh)

X-ray CT with the plugin for Core Imaging Library (CIL)

CIL
CORE IMAGING LIBRARY
ccpi.ac.uk/CIL



- **Set up forward model:**

```
A = FanBeam2DModel(det_count=560,
                   det_spacing=0.2,
                   angles=-np.linspace(0, 2*np.pi, 360),
                   source_object_dist=410.66,
                   object_detector_dist=143.08,
                   domain=(83.09, 83.09),
                   im_size=(500,500))
```

- **Load data as CUQIarray with Image2D geometry:**

```
y_obs = CUQIarray(sinogram, geometry=Image2D(im_shape=(360, 560)))
```

Riis et al. CUQIpy: I. computational uncertainty quantification for inverse problems in Python, Inverse Problems, https://doi.org/10.1088/1361-6420/ad22e7

X-ray CT with the plugin for Core Imaging Library (CIL)

- **Bayesian inverse problem:**

```
d = Gamma(1, 1e-4)
s = Gamma(1, 1e-4)
x = LMRF(1/d, geometry=A.domain_geometry)
y = Gaussian(A @ x, 1/s)
```

- **Specify posterior incl. observed data**

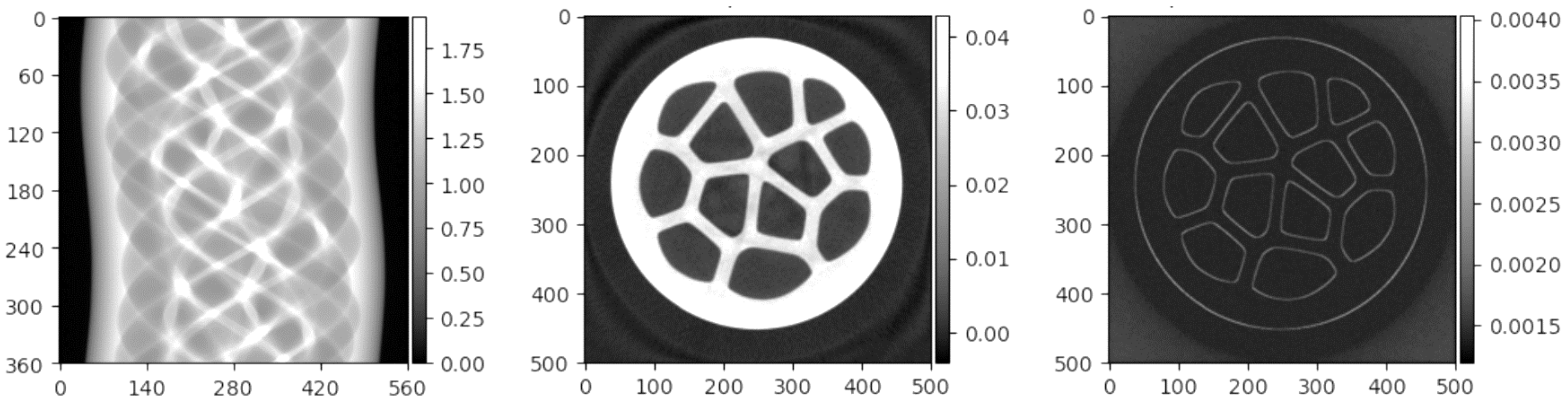
```
posterior = JointDistribution(d, s, x, y)(y=y_obs)
```

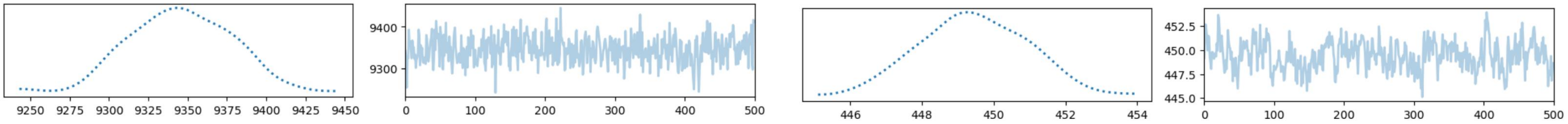
- **Gibbs sampling, exploiting conjugacy:**

```
sampling_strategy = {'d': ConjugateApprox,
                     's': Conjugate,
                     'x': UGLA}
samples = Gibbs(posterior, sampling_strategy).sample(500, 100)
```

Riis et al. CUQIpy: I. computational uncertainty quantification for inverse problems in Python, Inverse Problems, https://doi.org/10.1088/1361-6420/ad22e7

X-ray CT with the plugin for Core Imaging Library (CIL)

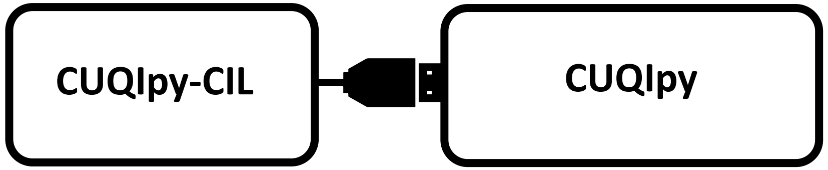




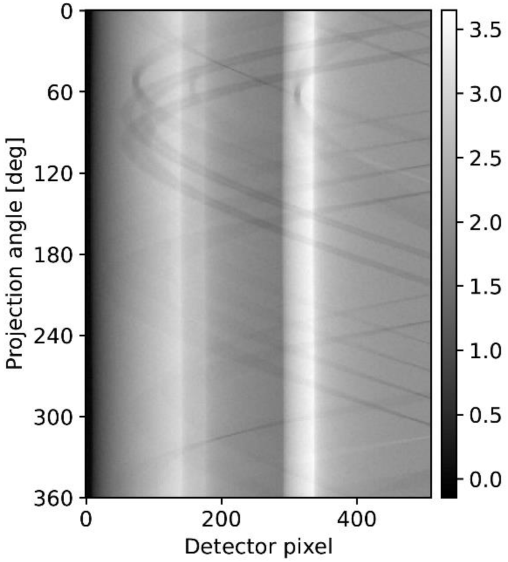
(a) Data precision s (b) Prior precision d

Riis et al. CUQIpy: I. computational uncertainty quantification for inverse problems in Python, Inverse Problems, https://doi.org/10.1088/1361-6420/ad22e7

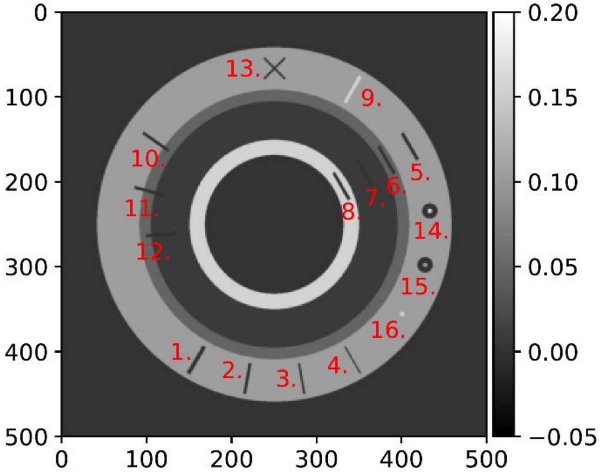
Case: Defect Detection in Subsea Pipes



- Crucial for oil and gas transport
- Non-invasive testing: CT



sinogram



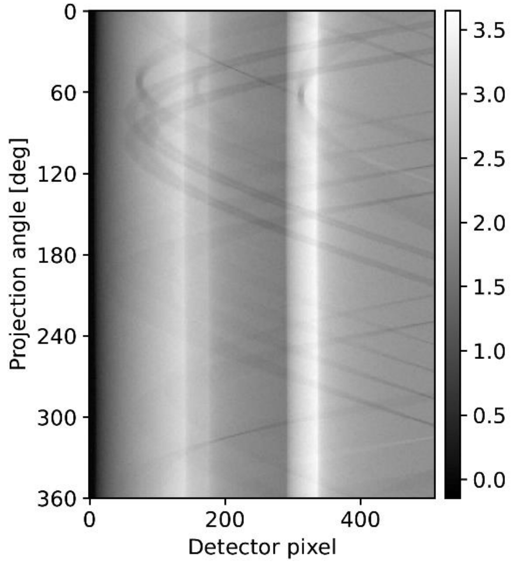
phantom

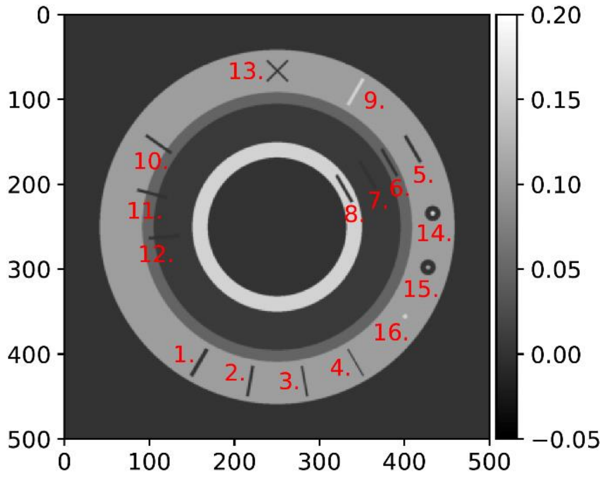
Christensen, Uribe, Riis and Jørgensen: Structural Gaussian priors for Bayesian CT reconstruction of subsea pipes, AMSE, 31, 1, 2023 (doi.org/10.1080/27690911.2023.2224918)

Christensen, Riis, Pereyra and Jørgensen: A Bayesian approach for CT reconstruction with defect detection for subsea pipelines, Inverse Problems, 40, 025003, https://doi.org/10.1088/1361-6420/ad1348

Case: Defect Detection in Subsea Pipes
Forward model

- Separation of pipe and defect
- Defect uncertainty quantification
- Likelihood of pixels being defect (positive or negative)
- Implemented using Gibbs sampling from CUQIpy





Forward model

$$\mathbf{b} = \mathbf{A}(\mathbf{z} + \mathbf{d}) = \mathbf{A}\mathbf{z} + \mathbf{A}\mathbf{d}$$

Pipe structure **Small defects**

```
A@z + A@d
```

Christensen, Uribe, Riis and Jørgensen: Structural Gaussian priors for Bayesian CT reconstruction of subsea pipes, AMSE, 31, 1, 2023 (doi.org/10.1080/27690911.2023.2224918)

Christensen, Riis, Pereyra and Jørgensen: A Bayesian approach for CT reconstruction with defect detection for subsea pipelines, Inverse Problems, 40, 025003, https://doi.org/10.1088/1361-6420/ad1348

Case: Defect Detection in Subsea Pipes
Bayesian model

Likelihood

$$\mathbf{b} \mid \mathbf{z}, \mathbf{d} \sim \mathcal{N}(\mathbf{A}\mathbf{z} + \mathbf{A}\mathbf{d}, \lambda^{-1}\mathbf{I})$$

```
b = Gaussian(A@z + A@d, 1/lambda_noise)
```

Pipe Prior z: Structural Gaussian

$$\mathbf{z} \sim \mathcal{N}\left(\boldsymbol{\mu}_{SGP}, \left(\mathbf{R}_{SGP}^T \mathbf{R}_{SGP}\right)^{-1}\right)$$

```
z = JointGaussianSqrtPrec(mean=[mu1, mu2, …],
    prec=[R0, R1, …], … )
```

Defect Prior d: Sparsity + spatially coherent Gamma MRF

$$\mathbf{d} \mid \mathbf{s} \sim \mathcal{N}\left(0, f_d(\mathbf{s})\right)$$
$$\mathbf{s} \mid \mathbf{w} \sim \mathcal{IG}\left(\omega_0, f_s(\mathbf{w})\right)$$
$$\mathbf{w} \mid \mathbf{s} \sim \mathcal{G}\left(\omega_0, f_w(\mathbf{s})\right)$$

```
d = Gaussian(0, f_d(s))
s = InverseGamma(w0, f_s(w))
w = Gamma(w0, f_w(s))
```

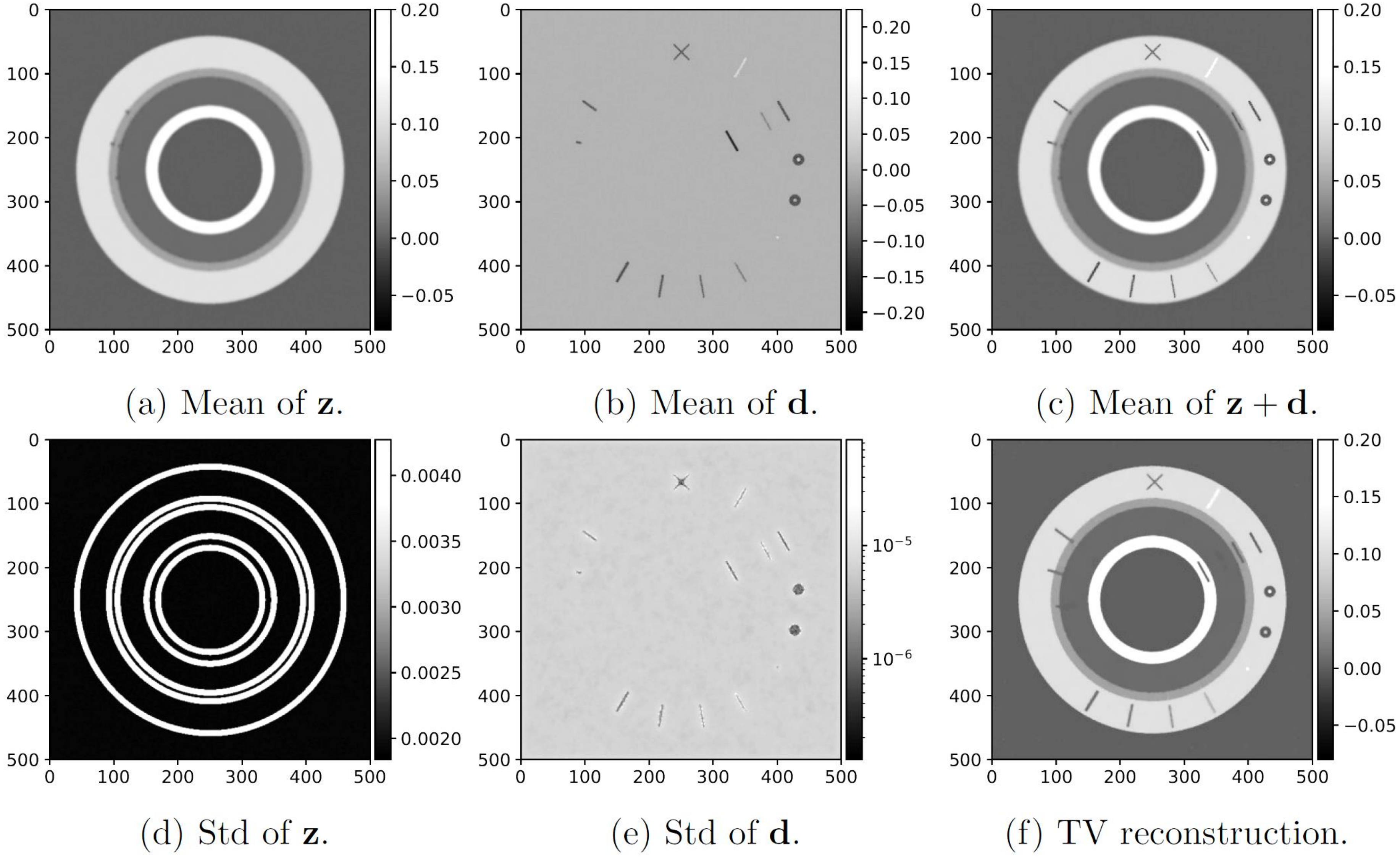
Posterior

$$p(\mathbf{z}, \mathbf{d}, \mathbf{s}, \mathbf{w} \mid \mathbf{b}) \propto p(\mathbf{b} \mid \mathbf{z}, \mathbf{d}) p(\mathbf{z}) p(\mathbf{d} \mid \mathbf{s}) p(\mathbf{s}, \mathbf{w})$$

```
post = JointDistribution(b, z, d, s, w)(b=b_data)
```

Christensen, Riis, Pereyra and Jørgensen: A Bayesian approach for CT reconstruction with defect detection for subsea pipelines, Inverse Problems, 40, 025003, https://doi.org/10.1088/1361-6420/ad1348

Case: Defect Detection in Subsea Pipes Reconstruction



(a) Mean of $\mathbf{z}$. (b) Mean of $\mathbf{d}$. (c) Mean of $\mathbf{z}+\mathbf{d}$.

(d) Std of $\mathbf{z}$. (e) Std of $\mathbf{d}$. (f) TV reconstruction.

Christensen, Riis, Pereyra and Jørgensen: A Bayesian approach for CT reconstruction with defect detection for subsea pipelines, Inverse Problems, 40, 025003, https://doi.org/10.1088/1361-6420/ad1348

Components of CUQIpy – Samplers

Sampler name	Description
`MetropolisHastings`	Metropolis–Hastings
`pCN`	preconditioned Crank–Nicolson
`ULA`	Unadjusted Langevin algorithm
`MALA`	Metropolis-Adjusted Langevin algorithm
`NUTS`	No U-Turn Sampler
`LinearRTO`	Linear Randomize-Then-Optimize
`UGLA`	Unadjusted Laplace Approximation
`Gibbs`	Gibbs sampler for joint distributions
`CWMH`	Component-Wise Metropolis–Hastings
`Conjugate`	Conjugate sampler
`ConjugateApprox`	Approximate conjugate sampler

Riis et al. CUQIpy: I. computational uncertainty quantification for inverse problems in Python, Inverse Problems, https://doi.org/10.1088/1361-6420/ad22e7

Components of CUQIpy – Test Problems

Test problem name	Description
`Deconvolution1D`	1D signal deblurring
`Deconvolution2D`	2D image deblurring
`Abel1D`	Rotationally symmetric computed tomography
`WangCubic`	Problem with nonlinear two-parameter forward model
`Heat1D`	Discrete Heat problem (time-dependent linear PDE)
`Poisson1D`	Discrete 1D Poisson problem (steady-state linear PDE)
`ParallelBeam2D`	2D parallel-beam CT using CIL

Riis et al. CUQIpy: I. computational uncertainty quantification for inverse problems in Python, Inverse Problems, https://doi.org/10.1088/1361-6420/ad22e7

Poisson PDE test problem with FEniCS in CUQIpy

$$\nabla \cdot \left(e^{w(\boldsymbol{\xi})} \nabla u(\boldsymbol{\xi})\right) = f(\boldsymbol{\xi}) \qquad \text{for} \qquad \boldsymbol{\xi} \in \Gamma = (0,1)^2 \tag{1}$$

written here in terms of the log-conductivity field, i.e., $w(\boldsymbol{\xi}) = \log \sigma(\boldsymbol{\xi})$ to ensure positivity of the inferred conductivity field. In this example, we assume zero boundary conditions on the left and right boundaries of the square domain and zero Neumann boundary conditions on the top and bottom boundaries; and a source term $f(\boldsymbol{\xi}) = 1$.

In CUQIpy we consider the discretized form of this problem,

$$\boldsymbol{y} = \boldsymbol{A}(\boldsymbol{x}), \tag{2}$$

where $\boldsymbol{A}$ is a nonlinear forward model, which corresponds to solving the discretized PDE to produce the observation $\boldsymbol{y}$ from a log-conductivity given in terms of a parameter $\boldsymbol{x}$.

```
A = FEniCSPoisson2D(dim=(32,32), field_type="KL", ...).model
```

Alghamdi et al. CUQIpy: II. computational uncertainty quantification for PDE-based inverse problems in Python, Inverse Problems, https://doi.org/10.1088/1361-6420/ad22e8

Specifying and solving Bayesian formulation

$$\boldsymbol{x} \sim \text{Gaussian}(\mathbf{0}, \boldsymbol{I})$$

$$\boldsymbol{y} \sim \text{Gaussian}(\boldsymbol{A}(\boldsymbol{x}), s_{\text{noise}}^2 \boldsymbol{I}),$$

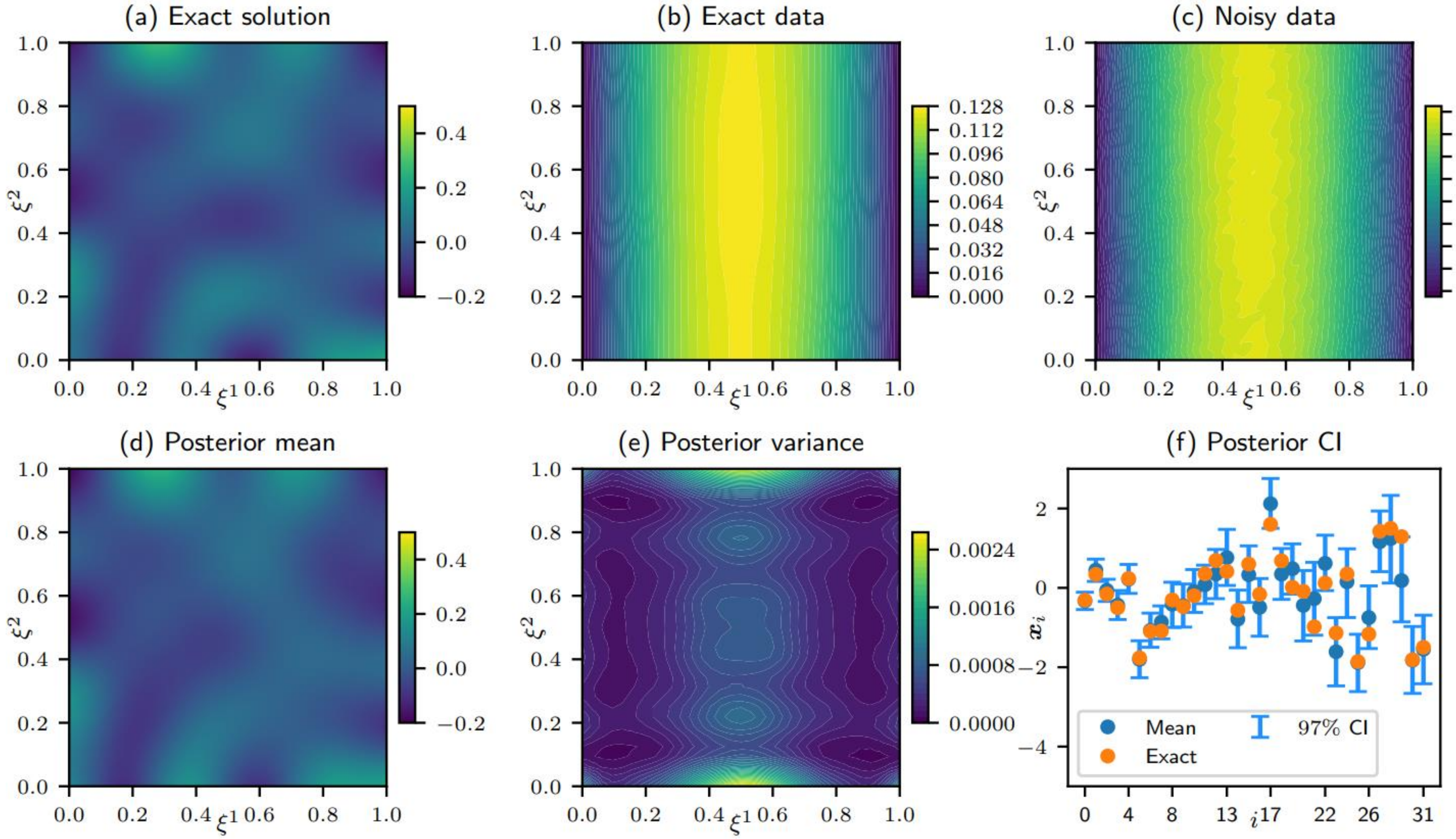
```
x = Gaussian(np.zeros(n_KL), 1, geometry=G_KL)
y = Gaussian(A(x), s_noise**2, geometry=G_FEM)
```

```
x_true = x.sample()
x_true.plot()
```

```
y_obs = y(x=x_true).sample()
y_obs.plot()
```

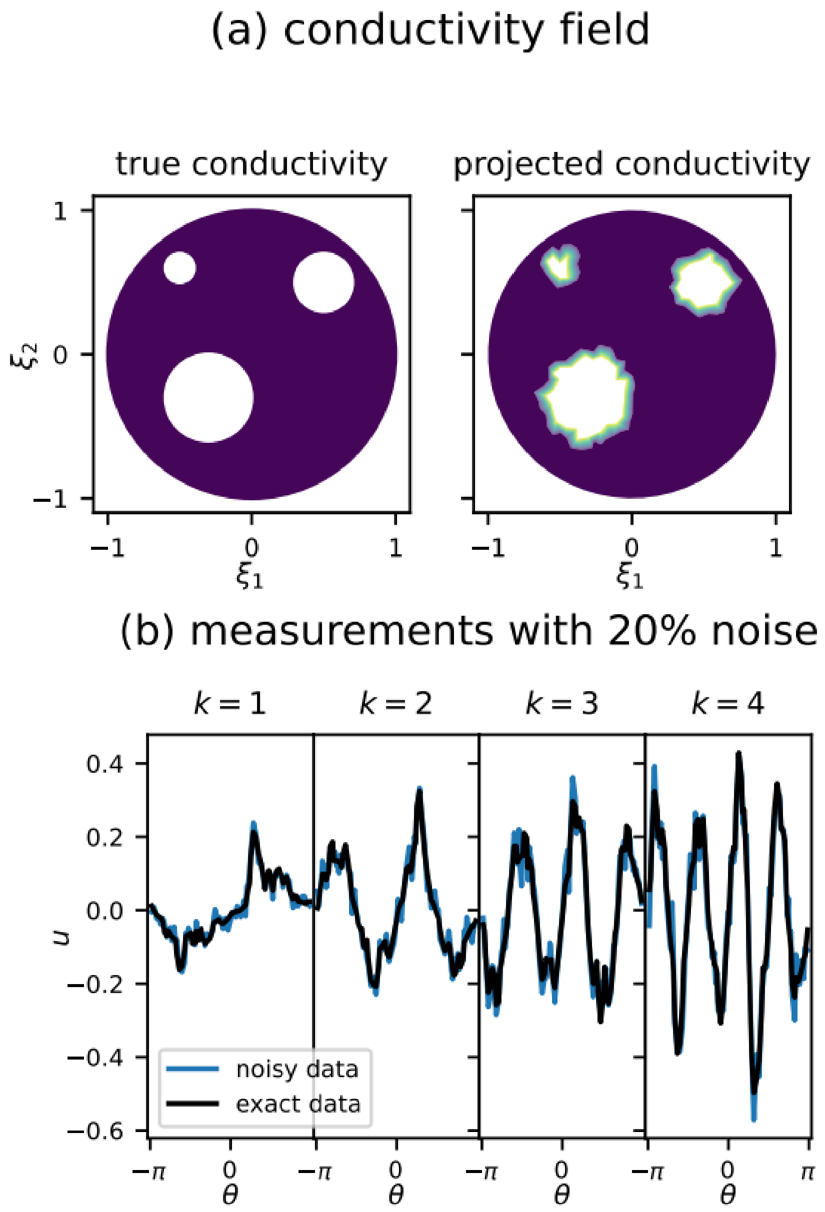
```
BP = BayesianProblem(y, x).set_data(y=y_obs)
BP.UQ()
```

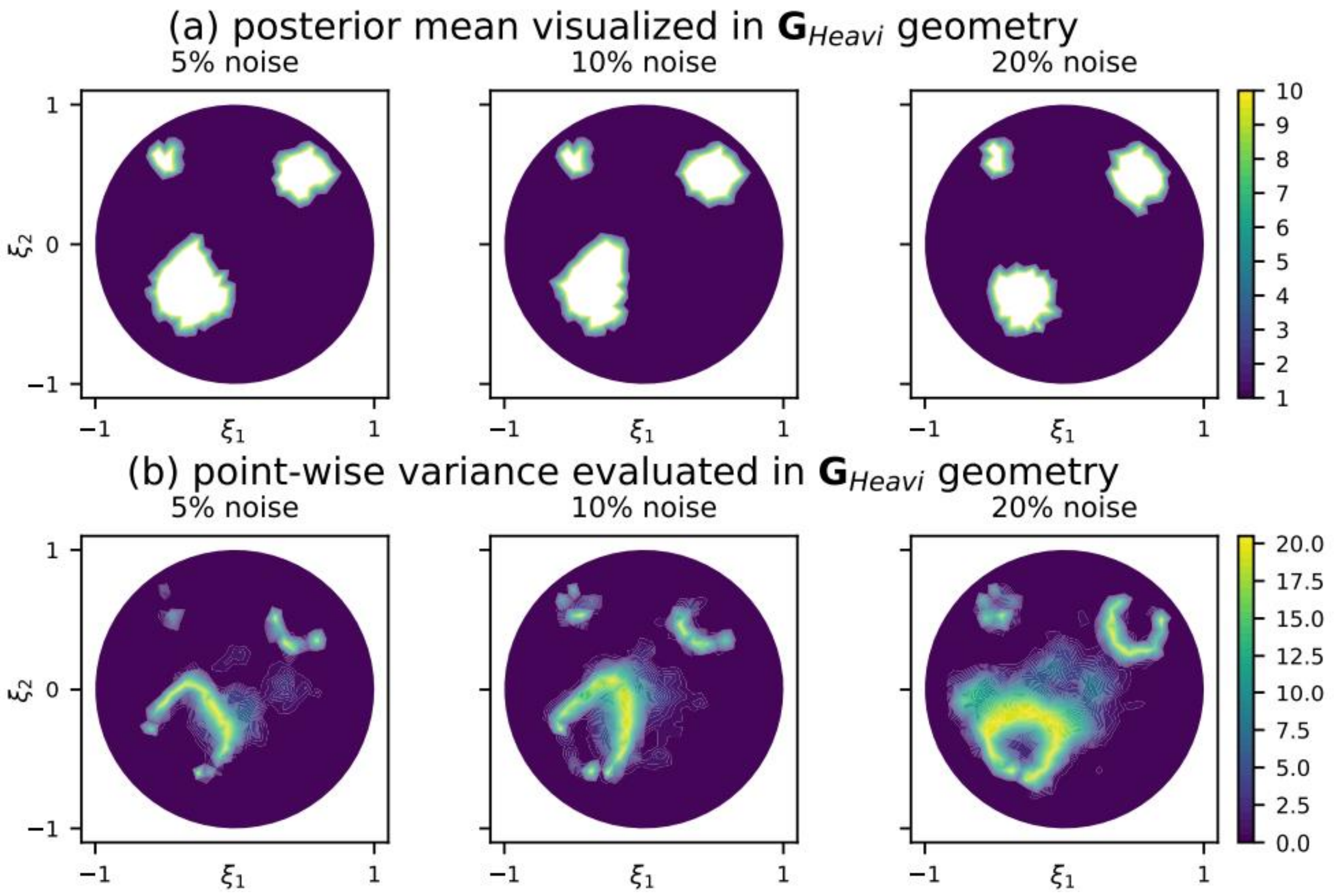
Alghamdi et al. CUQIpy: II. computational uncertainty quantification for PDE-based inverse problems in Python, Inverse Problems, https://doi.org/10.1088/1361-6420/ad22e8



Alghamdi et al. CUQIpy: II. computational uncertainty quantification for PDE-based inverse problems in Python, Inverse Problems, https://doi.org/10.1088/1361-6420/ad22e8

Electrical Impedance Tomography with CUQIpy





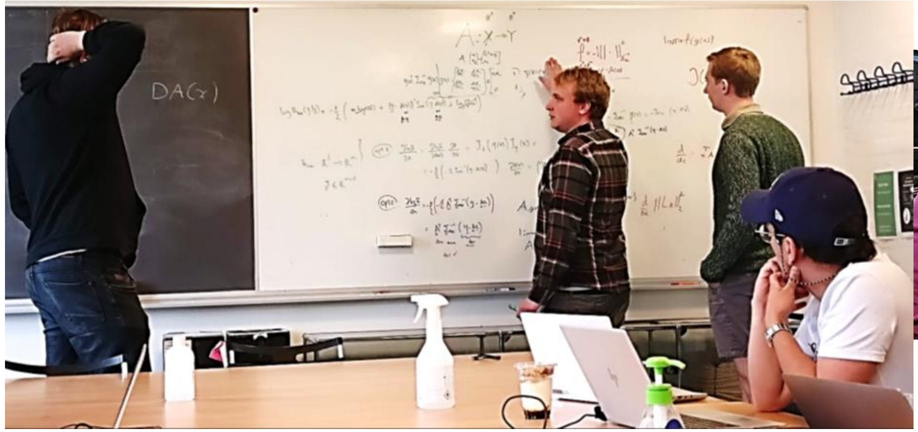
Alghamdi et al. CUQIpy: II. computational uncertainty quantification for PDE-based inverse problems in Python, Inverse Problems, https://doi.org/10.1088/1361-6420/ad22e8

VILLUM FONDEN

Collaborative Development

Involvement of CUQI team

- Feature requests
- Test cases
- Code contributions
- Use in teaching
- CUQIpy hackathons



Internal and user-facing CUQIpy hackathons

User training and hackathons

CUQIpy training @ IUQ workshop, Denmark, Sept 2022

"Well documented and easy to use."

"I think the whole user-experience was very smooth [...]"

"It's obvious that it is aimed towards non-experts, but it's also great that experts can really take advantage of the package and do more complex stuff."

CUQIpy User Showcase https://github.com/CUQI-DTU/CUQIpy-User-Showcase

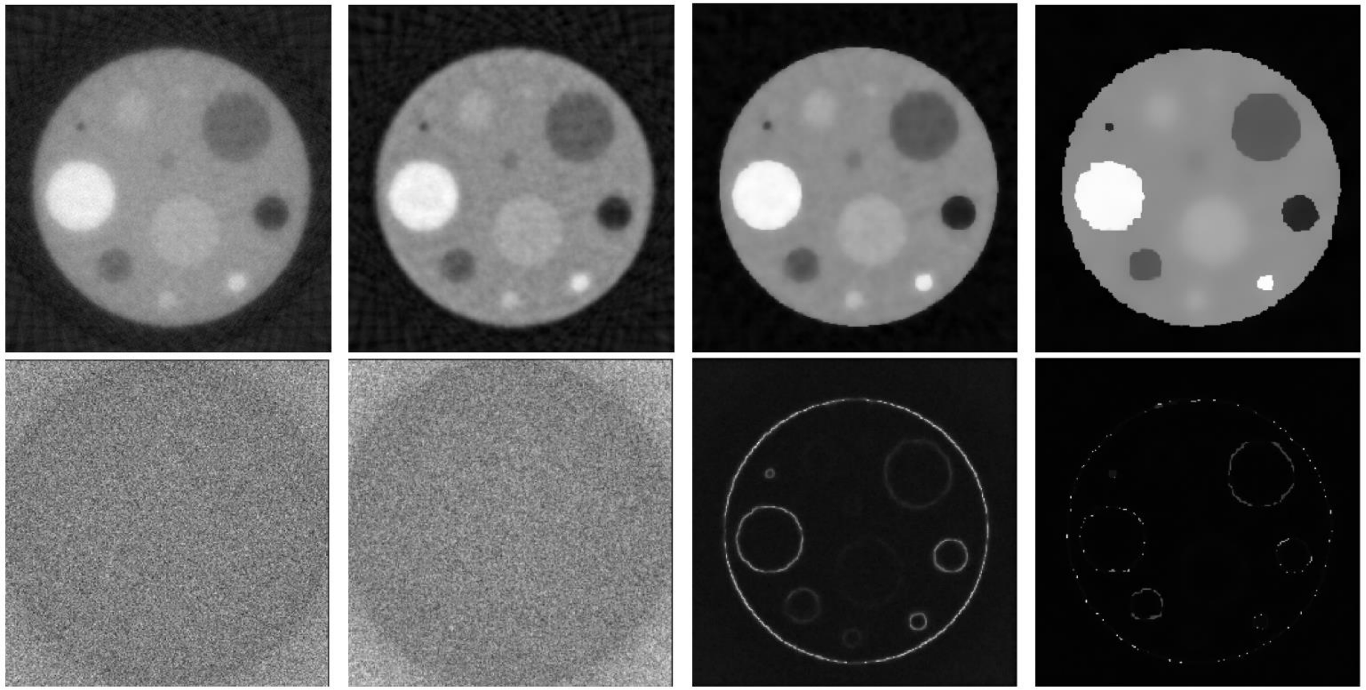
List of notebooks

- 001 Besov 2D deblurring *custom prior, deconvolution, gradient-based*
- 002 Eigenvalue forward *custom forward, custom sampler, independence sampler*
- 003 Error model *deconvolution*
- 004 Besov 1D deconvolution *custom prior, deconvolution, gradient-based*
- 005 Implicit priors *implicit priors, regularized Gaussain deconvolution, linearRTO*
- 006 TBA *Implicit “plug and play” prior*
- 007 TBA *super resolution, Laplace prior*
- 008 Delayed acceptance *custom forward, custom sampler, ODE*
- 009 Inverse Robin *custom forward, finite element, gradient_based*
- *Spring stiffness system: custom nonlinear forward*

CUQIpy + UM-Bridge

- 3 UM-Bridge benchmarks using CUQIpy
- Support for both client (UQ software) and server (numerical model) usage

CT benchmark



https://arxiv.org/abs/2402.13768

Democratizing Uncertainty Quantification

Linus Seelinger, Anne Reinarz, Mikkel B. Lykkegaard, Amal Mohammed A. Alghamdi, David Aristoff, Wolfgang Bangerth, Jean Bénézech, Matteo Diez, Kurt Frey, John D. Jakeman, Jakob Sauer Jørgensen, Ki-Tae Kim, Massimiliano Martinelli, Matthew Parno, Riccardo Pellegrini, Noemi Petra, Nicolai A. B. Riis, Katherine Rosenfeld, Andrea Serani, Lorenzo Tamellini, Umberto Villa, Tim J. Dodwell, Robert Scheichl

CUQIpy Conclusions

- Python framework for computational UQ for (imaging) inverse problems
- Unified framework for problems with and without PDE-based forward model
- Collection of priors, samplers, test problems, …
- Hierarchical problems
- Exploit structure e.g. linearity, conjugacy as much as possible
- High-level modelling framework, automatic sampler selection
- Fully configurable

Advertisement: https://www.icms.org.uk/UQIPI24

UQIPI24: UQ for Inverse Problems and Imaging

16 - 20 Sep 2024

ICMS, Bayes Centre, Edinburgh

Open in google maps

This workshop will bring together specialists in UQ for inverse problems and imaging, and we invite talks related to the development of **theory**, **methodology**, and **software**. We also invite talks about interesting **applications** of UQ in imaging. The goal is to stimulate networking and collaboration between researchers and students in these areas, and to present state-of-the-art research results.

Plenary Speakers

- **Yoann Altmann**, Heriot-Watt University
- **Tatiana Bubba**, University of Bath
- **Per Christian Hansen**, Technical University of Denmark
- **Aku Seppänen**, University of Eastern Finland
- **Julián Tachella**, CNRS and ENS de Lyon
- **Faouzi Triki**, Grenoble-Alpes University

	Monday	Tuesday	Wednesday	Thursday	Friday
Morning	UQ tutorial	CUQIpy course	Workshop	Workshop	Workshop
After-noon	CUQIpy course	Workshop	Workshop	Workshop	Social event
Evening	CUQIpy course for the nerds	Reception		Guided tour & workshop dinner	

CUQIpy

- **Install** pip install cuqipy

- **Website** cuqi-dtu.github.io/CUQIpy

- **Training material** github.com/CUQI-DTU/CUQIpy-demos

- **Expansion plugins**
  - X-ray CT github.com/CUQI-DTU/CUQIpy-CIL
  - PDE finite element github.com/CUQI-DTU/CUQIpy-FEniCS
  - PyTorch autodiff github.com/CUQI-DTU/CUQIpy-PyTorch

- **Publications**
  - Riis *et al.* (2024) https://doi.org/10.1088/1361-6420/ad22e7
  - Alghamdi *et al.* (2024) https://doi.org/10.1088/1361-6420/ad22e8

Thanks for your attention!