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

AIP 2023 | MS35 Edge-preserving UQ for imaging
Göttingen, Germany | 4 September 2023

Who? CUQIpy Funding, Developers and Contributors

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

Starting Oct. 2023

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.

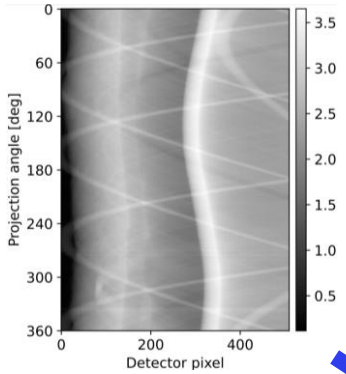
CUQIpy

DTU

CUQI VILLUM FONDEN

Forward model & data

$\mathbf{A}$ $\mathbf{b}$



Bayesian inverse problem

modelling framework

$$p(\mathbf{x}, \delta|\mathbf{b}) \propto p(\mathbf{b}|\mathbf{x})\, p(\mathbf{x}|\delta)\, p(\delta)$$

Posterior Likelihood Priors

Parameter assumptions

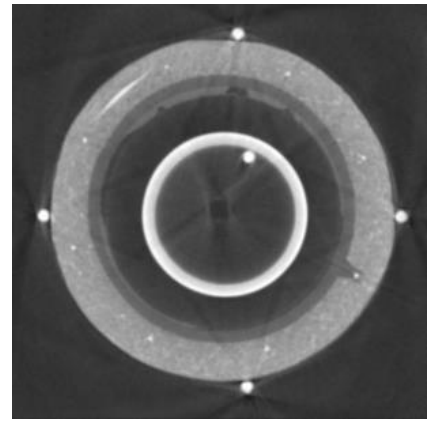
$$\mathbf{b} \sim \mathcal{N}(\mathbf{Ax}, \sigma^2_{\text{noise}}\mathbf{I})$$
$$\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \delta^2\mathbf{I})$$
$$\delta \sim \text{Gamma}(a, b)$$

Computational UQ engine

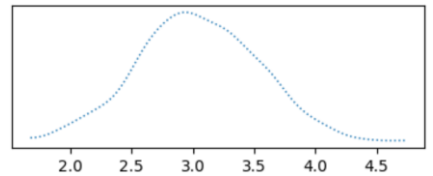
- Efficient posterior sampling using structure
- High-level interface for non-experts
- Full control for experts
- Hierarchical problem support
- Catalogue of test problems, priors, …

Parameter estimates

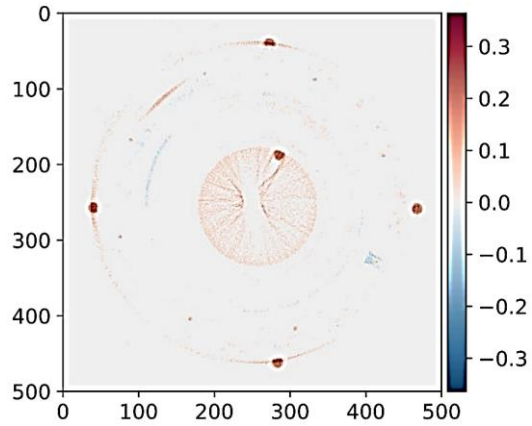
$\mathbf{x}$



δ



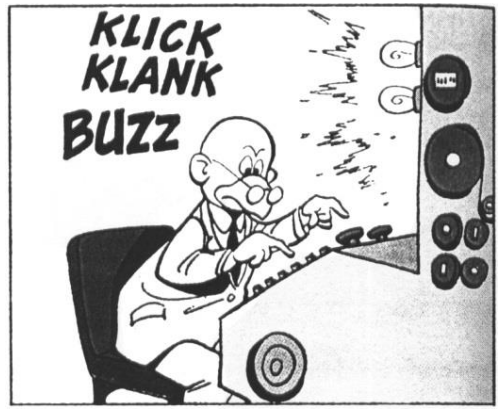
UQ information



CUQIpy in a Nutshell

Vision

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

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.

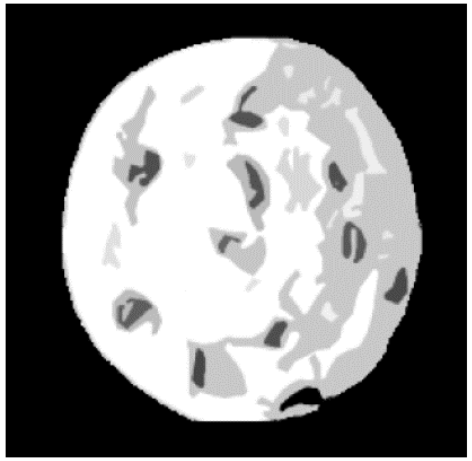
The niche that CUQIpy is aimed at:

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

Cookie deblurring with CUQIpy

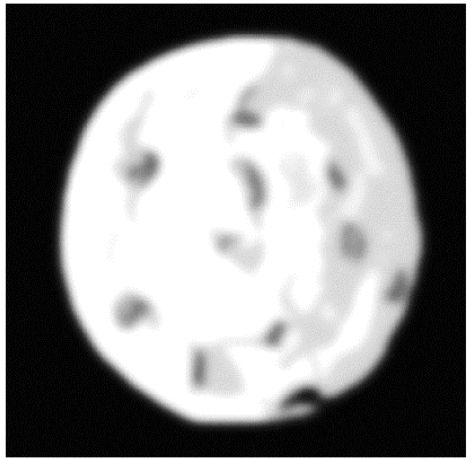
$$Ax = y$$

True



`info.exactSolution.plot()`

Blurred, noisy



`y_obs.plot()`

Using testproblem library

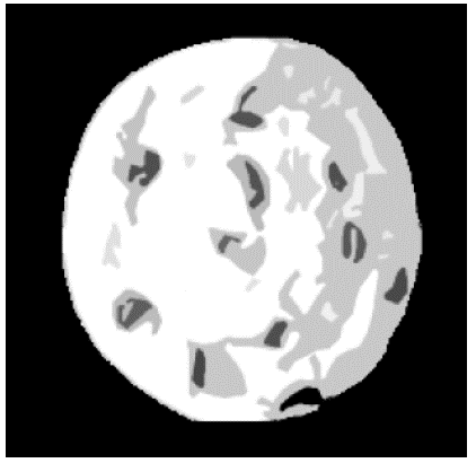
```
A, y_obs, info = Deconvolution2D(dim=512, phantom="cookie")
```

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

Cookie deblurring with CUQIpy

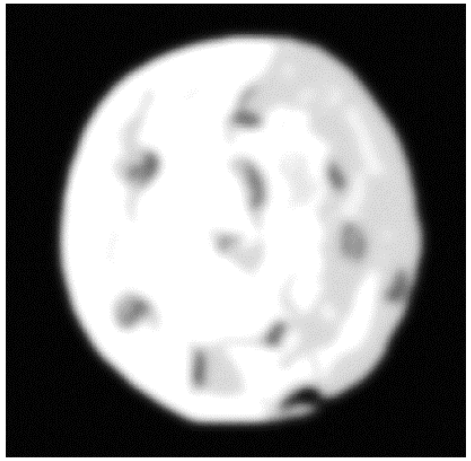
$$Ax = y$$

True



`info.exactSolution.plot()`

Blurred, noisy



`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

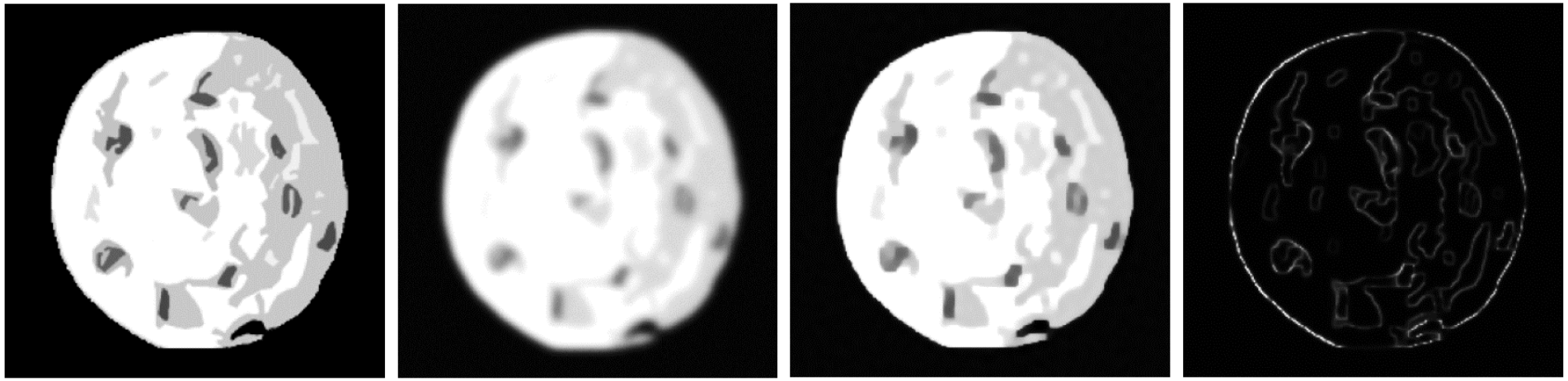
```
print(A)
```

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

Cookie deblurring with CUQIpy

$$Ax = y$$

True | Blurred, noisy | Posterior mean | Posterior std



$$d \sim \text{Gamma}(1, 10^{-4})$$

$$s \sim \text{Gamma}(1, 10^{-4})$$

$$\boldsymbol{x} \sim \text{LMRF}(d^{-1}),$$

$$\boldsymbol{y} \sim \text{Gaussian}(\boldsymbol{A}\boldsymbol{x}, s^{-1}\boldsymbol{I})$$

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()
```

Providing full control to expert users

Bayesian 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)

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

Inside BP posterior is defined:

$$p(\mathbf{x}, s, d \mid \mathbf{y}) \propto p(\mathbf{y} \mid \mathbf{x}, s)p(\mathbf{x} \mid d)p(s)p(d)$$

Uncertainty Quantificaiton

```
BP.UQ()
```

Providing full control to expert users

Bayesian 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)

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

Inside BP posterior is defined:

$$p(\mathbf{x}, s, d \mid \mathbf{y}) \propto p(\mathbf{y} \mid \mathbf{x}, s)p(\mathbf{x} \mid d)p(s)p(d)$$

Uncertainty Quantificaiton

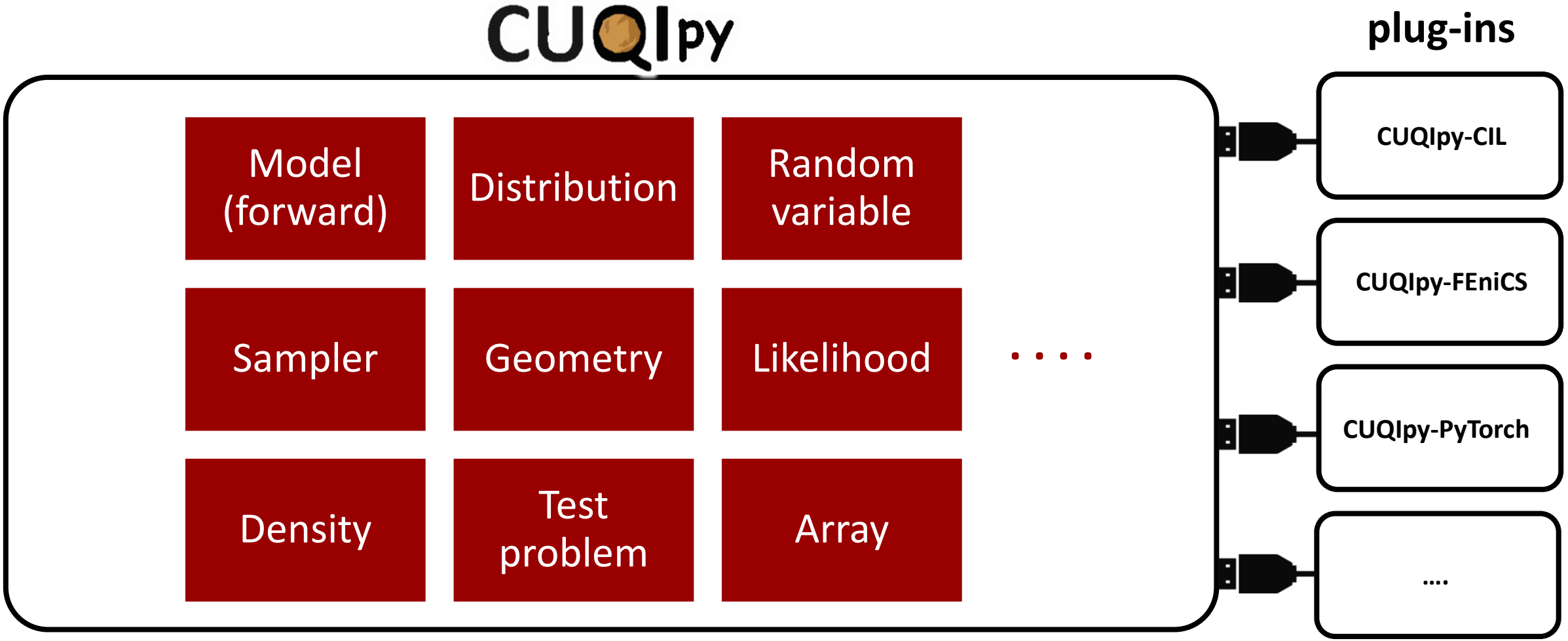
```
posterior = BP.posterior

sampling_strategy= {
  "x" : UGLA,
  ("d", "s") : Conjugate
}

sampler = Gibbs(posterior, sampling_strategy)

samples = sampler.sample(5000, 2000)
```

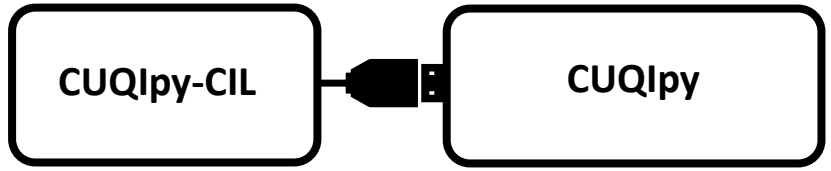
Overview of CUQIpy Modules and Plug-ins



CUQIpy: github.com/CUQI-DTU

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

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 CUQIarrey with Image2D geometry:**

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

Riis et al. *CUQIpy - Part I: computational uncertainty quantification for inverse problems in Python*, https://arxiv.org/abs/2305.16949

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)
```

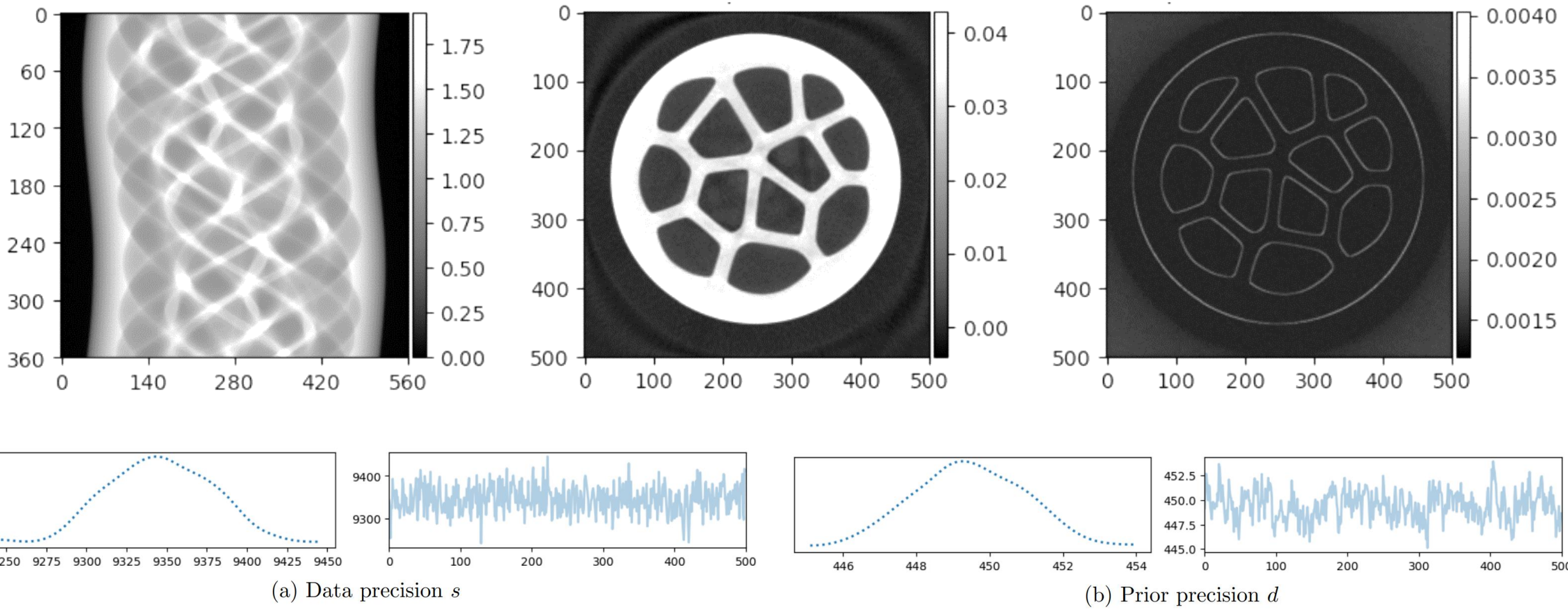
- **Specify posterior incl. observed data**

```
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)
```

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

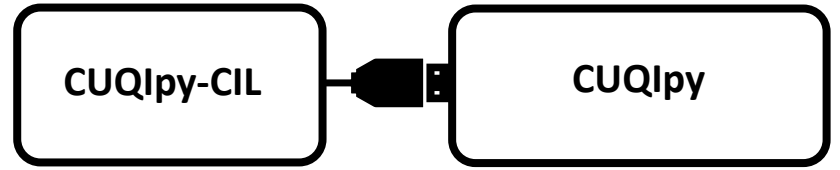


(a) Data precision s

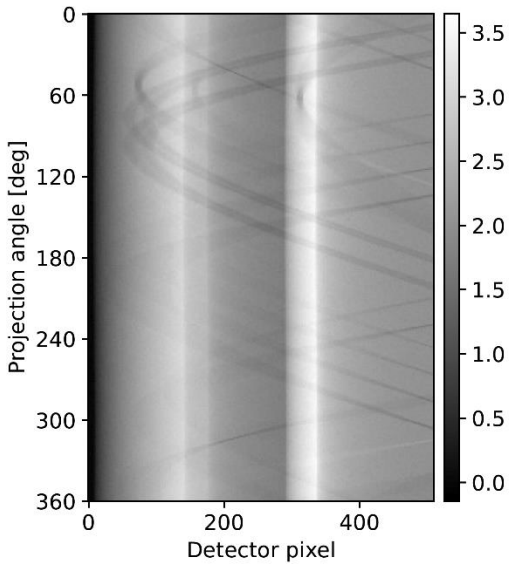
(b) Prior precision d

Riis et al. *CUQIpy - Part I: computational uncertainty quantification for inverse problems in Python*, https://arxiv.org/abs/2305.16949

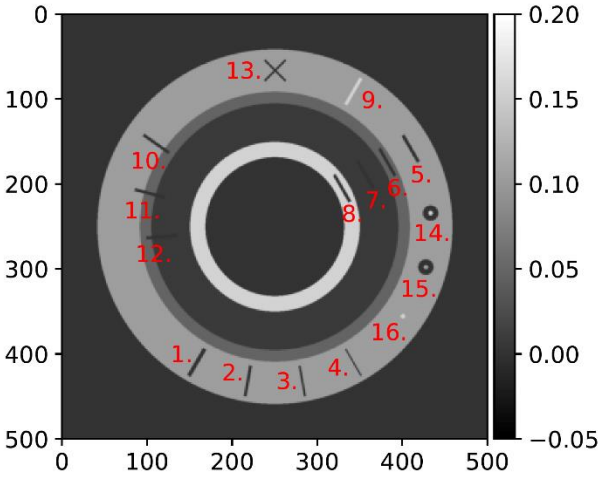
Case: Defect Detection in Subsea Pipes



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



sinogram



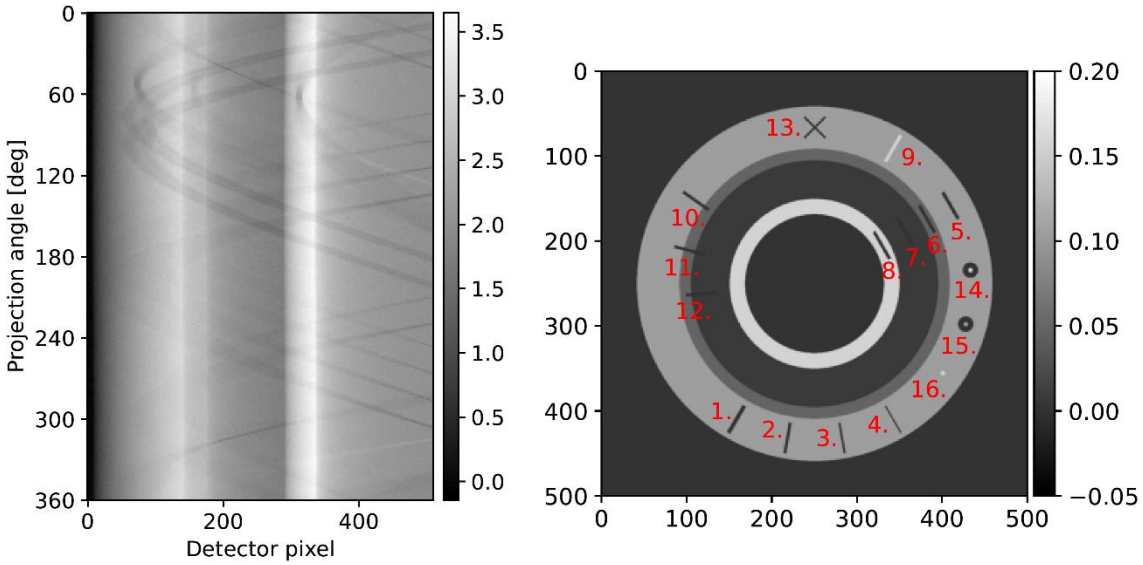
phantom

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 (in review)

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 (in review)

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

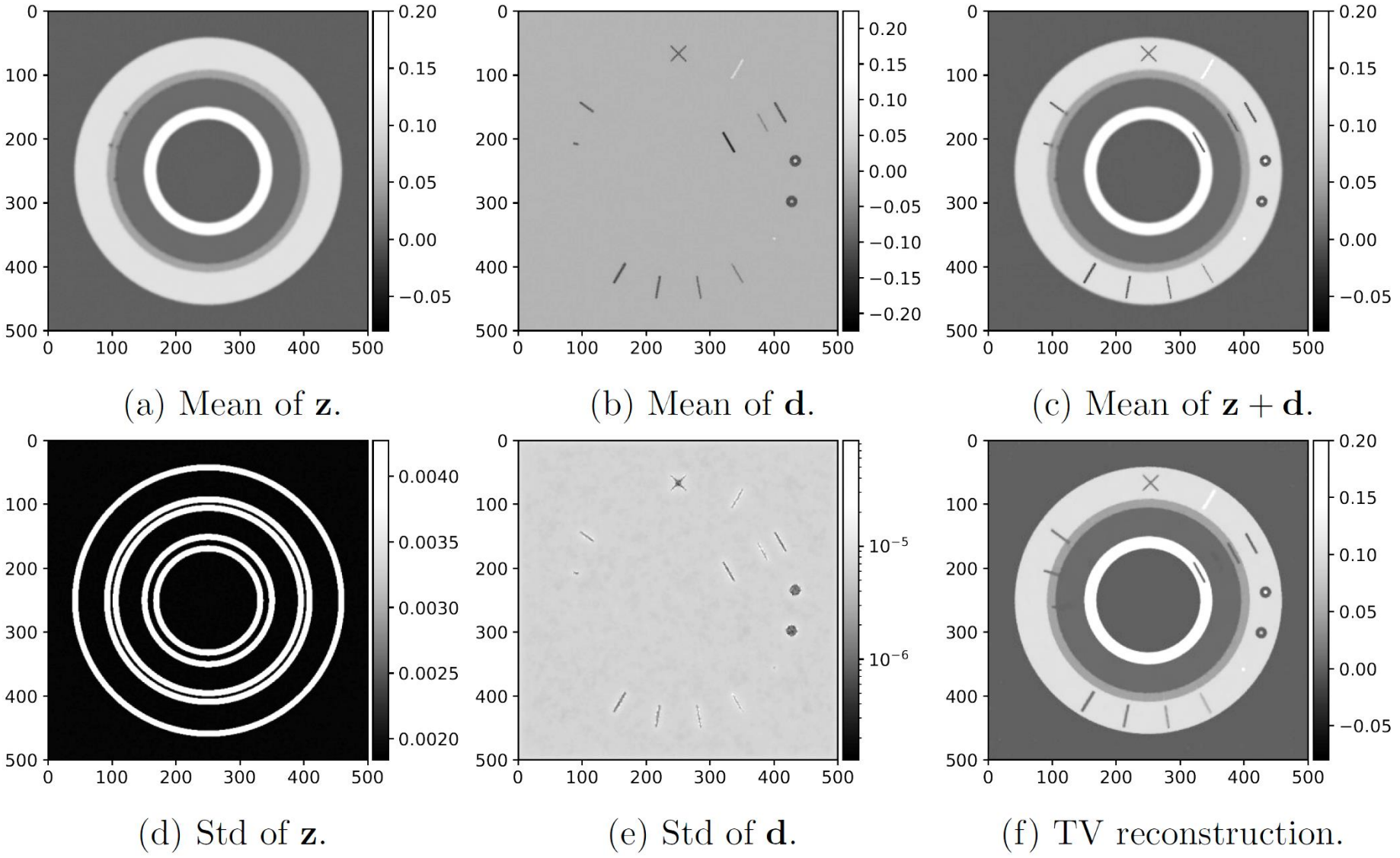
$$p(\mathbf{z}, \mathbf{d}, \mathbf{s}, \mathbf{w} \mid \mathbf{b}) \propto p(\mathbf{b} \mid \mathbf{z}, \mathbf{d}) p(\mathbf{z}) p(\mathbf{d} \mid \mathbf{s}) p(\mathbf{s}, \mathbf{w})$$

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

Christensen, 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 (in review)

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 (in review)

Components of CUQIpy – Distributions

Distribution name	Description
`Beta`	Beta distribution
`Cauchy`	Cauchy distribution
`Gamma`	Gamma distribution
`Gaussian`	Multivariate Gaussian distribution
`InverseGamma`	Inverse Gamma distribution
`Laplace`	Laplace distribution
`Lognormal`	Log-normal distribution
`Uniform`	Uniform Distribution
`CMRF`	Cauchy Markov Random Field
`GMRF`	Gaussian Markov Random Field
`JointGaussianSqrtPrec`	Gaussian of multiple square-root precision matrices
`LMRF`	Laplace Markov Random Field
`JointDistribution`	Joint distribution of multiple parameters
`Posterior`	Posterior distribution
`DistributionGallery`	Collection of benchmark distributions
`UserDefinedDistribution`	Defined by functions for `logpdf` and/or `sample`

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

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

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 - Part II: computational uncertainty quantification for PDE-based inverse problems in Python,* https://arxiv.org/abs/2305.16951

Specifying and solving Bayesian formulation

$$\boldsymbol{x} \sim \text{Gaussian}(\boldsymbol{0}, \boldsymbol{I})$$
$$\boldsymbol{y} \sim \text{Gaussian}(\boldsymbol{A}(\boldsymbol{x}), s_{\text{noise}}^2 \boldsymbol{I}),$$

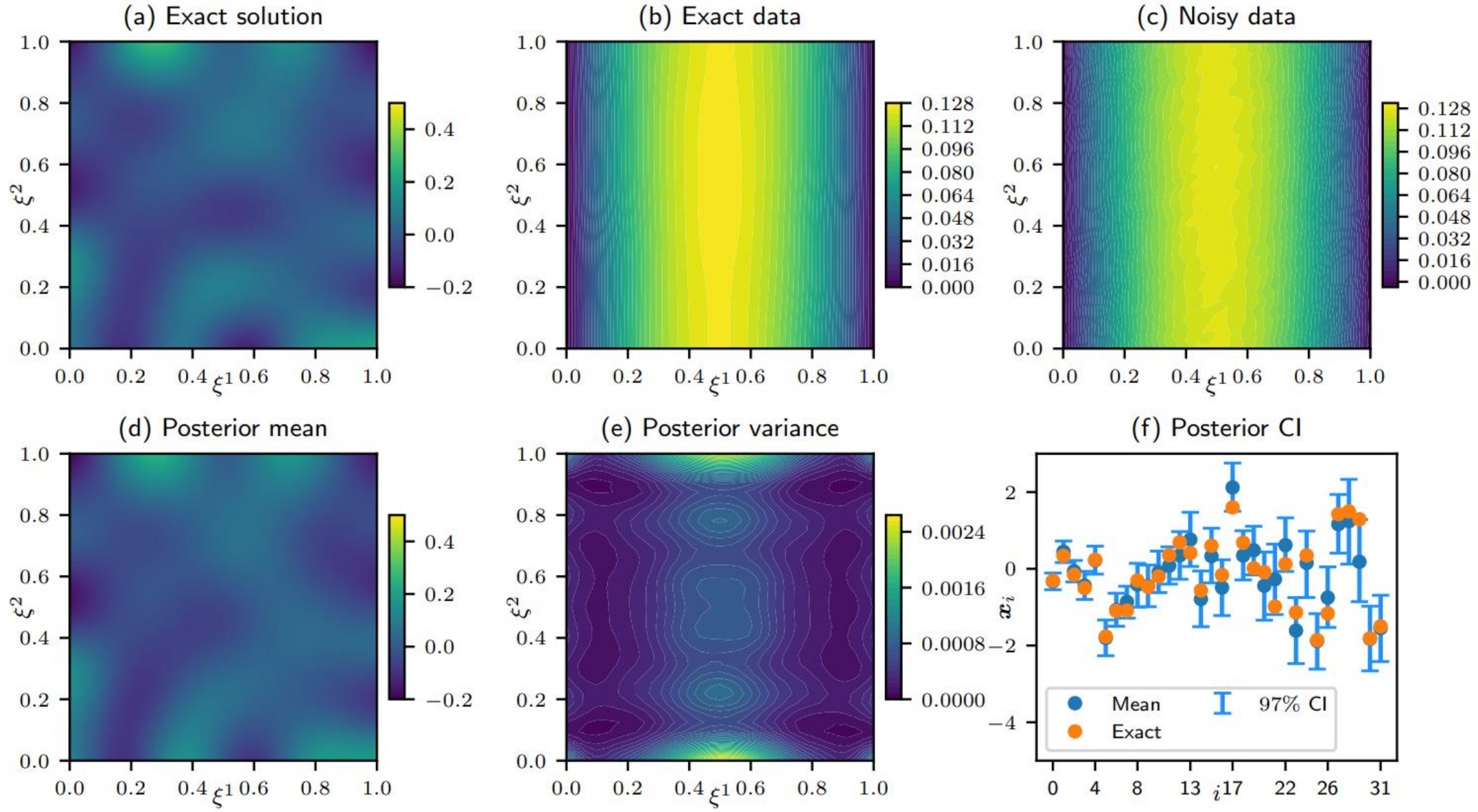
```
x = Gaussian(np.zeros(n_KL), 1, geometry=G_KL)
y = Gaussian(A(x), s_noise**2, geometry=G_FEM)
```

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

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

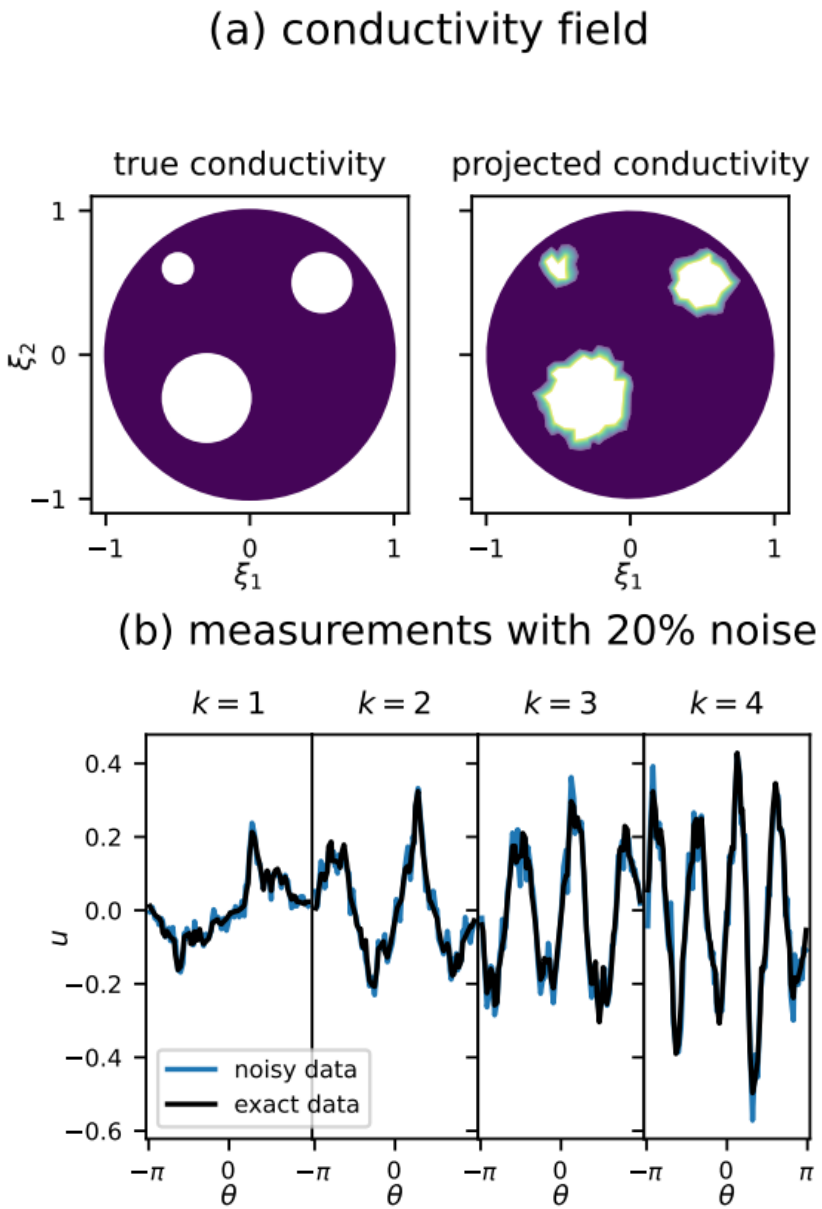
```
BP = BayesianProblem(y, x).set_data(y=y_obs)
BP.UQ()
```

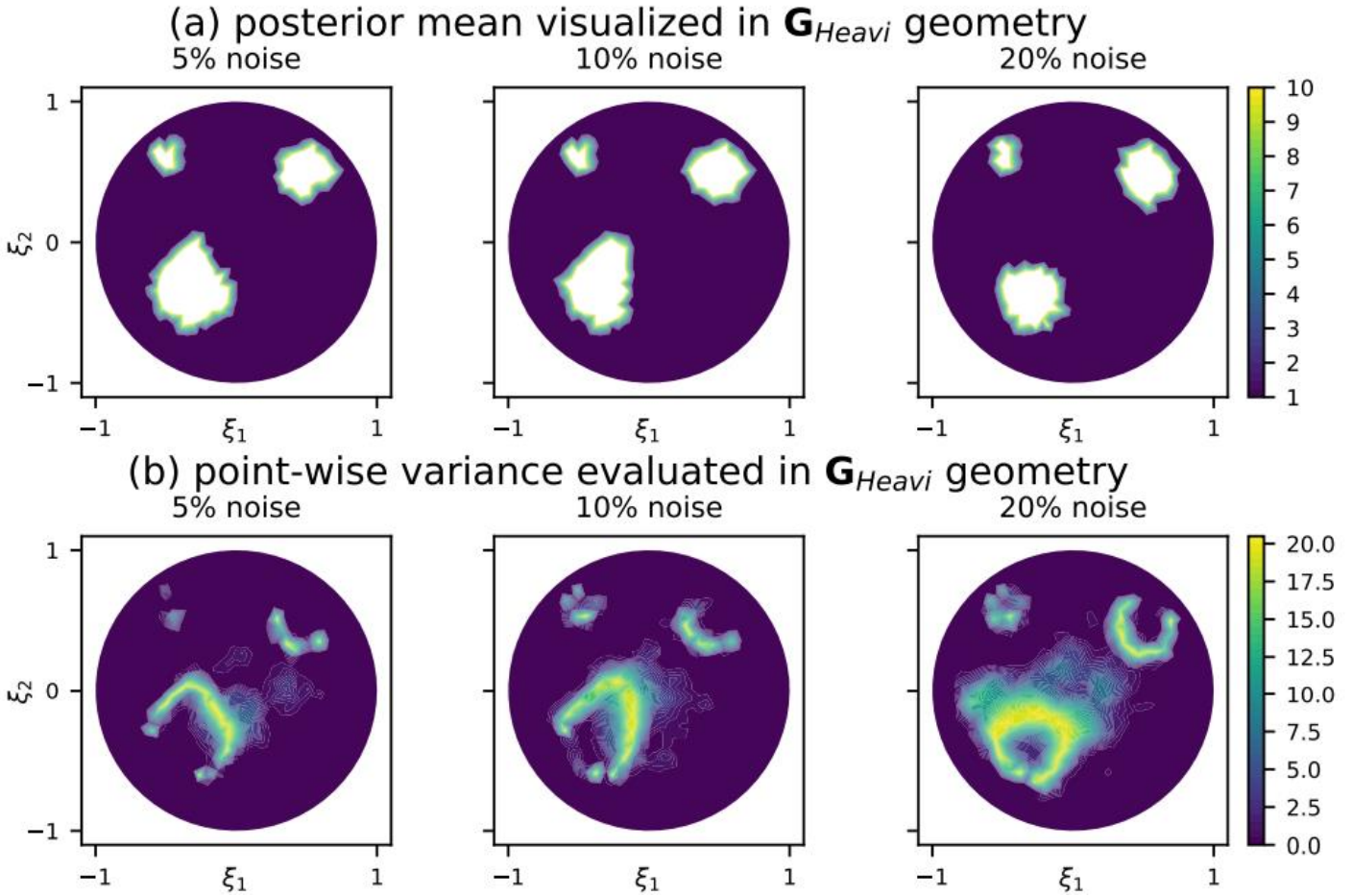
Alghamdi et al. *CUQIpy - Part II: computational uncertainty quantification for PDE-based inverse problems in Python,* https://arxiv.org/abs/2305.16951



Alghamdi et al. *CUQIpy - Part II: computational uncertainty quantification for PDE-based inverse problems in Python,* https://arxiv.org/abs/2305.16951

Electrical Impedance Tomography with CUQIpy



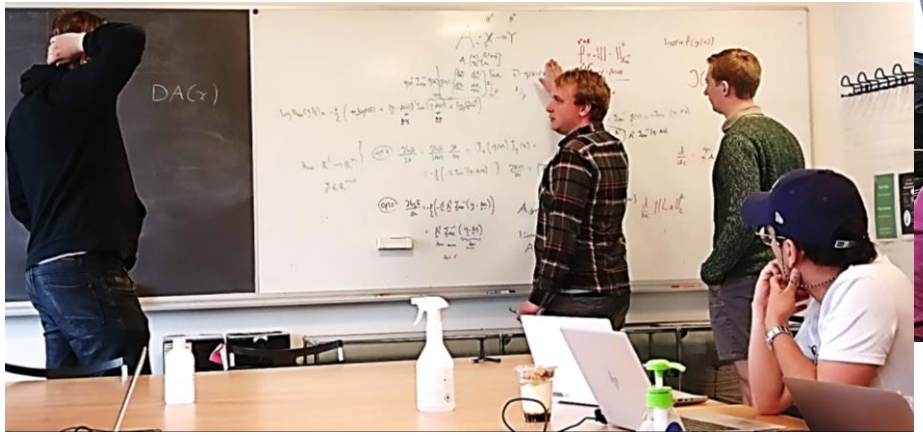


Alghamdi et al. *CUQIpy - Part II: computational uncertainty quantification for PDE-based inverse problems in Python,* https://arxiv.org/abs/2305.16951

Collaborative Development

Involvement of CUQI team

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



Internal CUQI hackathons

CUQIpy Training

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 Conclusions

- Python framework for computational UQ for 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 of other talks at AIP 2023

MS36 1 + 2: Advances in limited-data X-ray tomography – Thursday PM

- Edoardo Pasca: *Directional regularization with the Core Imaging Library for limited-angle CT in the Helsinki Tomography Challenge 2022*
- JJ: *Bayesian approach to limited-data CT reconstruction for inspection of subsea pipes*

MS51 1 + 2: Analysis, numerical computation, and uncertainty quantification for stochastic PDE-based inverse problems – Thursday PM

- Amal Alghamdi: *Optimization under uncertainty for the Helmholtz equation with application to photonic nanojets configuration*
- Several other CUQI members

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
- **Preprints on arxiv**

[Submitted on 26 May 2023]

CUQIpy -- Part I: computational uncertainty quantification for inverse problems in Python

Nicolai A B Riis, Amal M A Alghamdi, Felipe Uribe, Silja L Christensen, Babak M Afkham, Per Christian Hansen, Jakob S Jørgensen arxiv.org/abs/2305.16949

[Submitted on 26 May 2023]

CUQIpy -- Part II: computational uncertainty quantification for PDE-based inverse problems in Python

Amal M A Alghamdi, Nicolai A B Riis, Babak M Afkham, Felipe Uribe, Silja L Christensen, Per Christian Hansen, Jakob S Jørgensen arxiv.org/abs/2305.16951