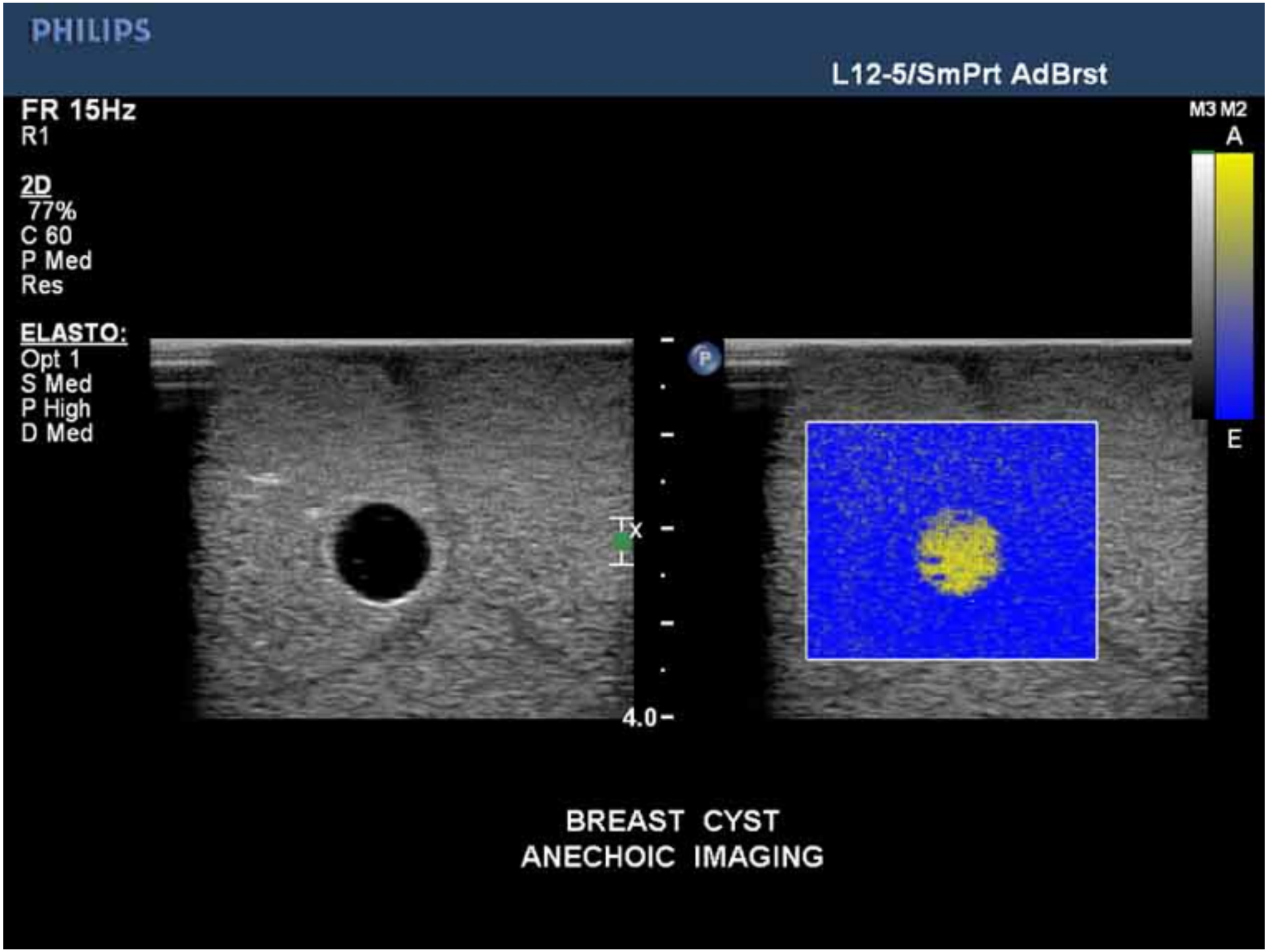
A Bayesian inversion approach to recovering material parameters in hyperelastic solids using dolfin-adjoint

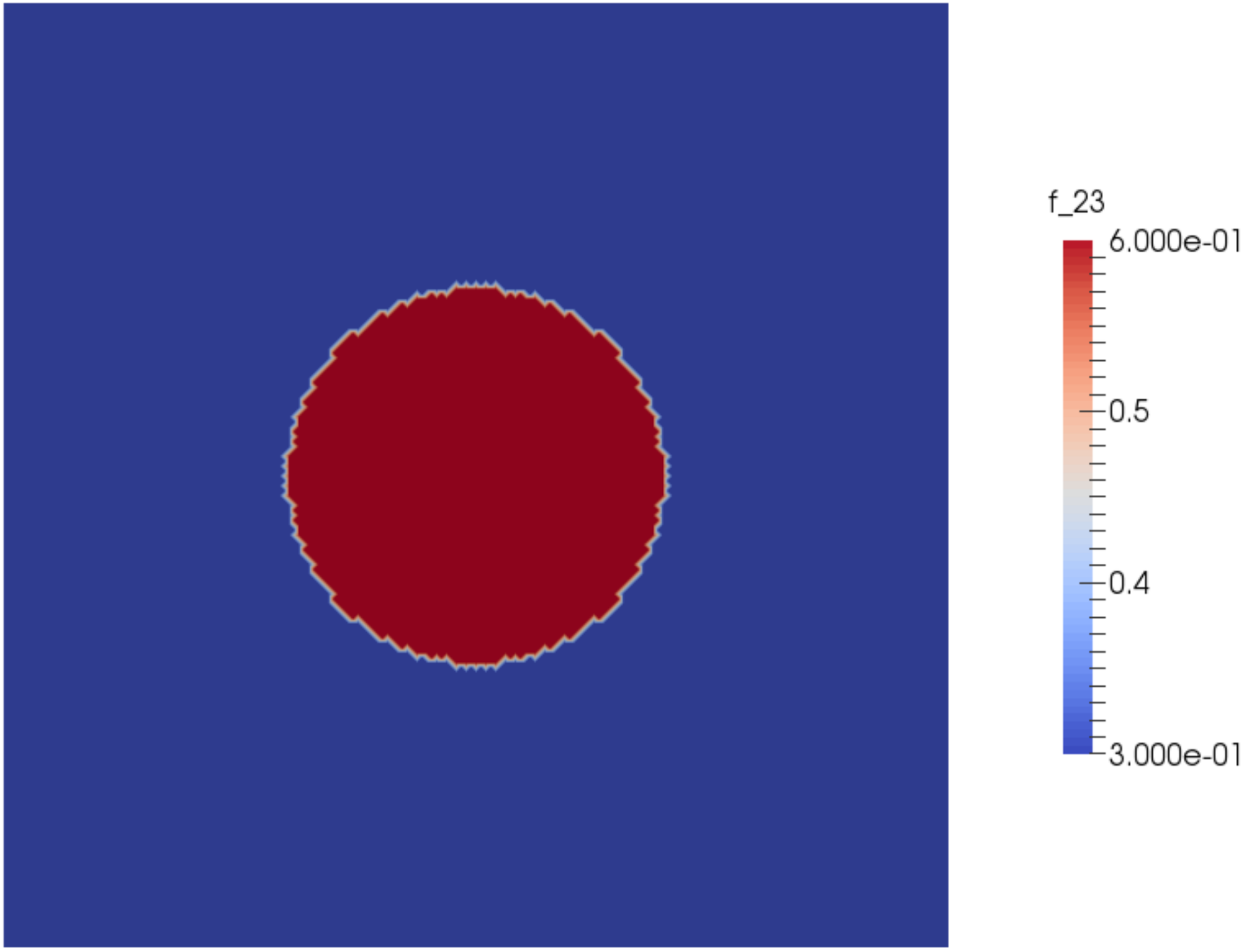
Jack S. Hale, Patrick E. Farrell, Stéphane P. A. Bordas

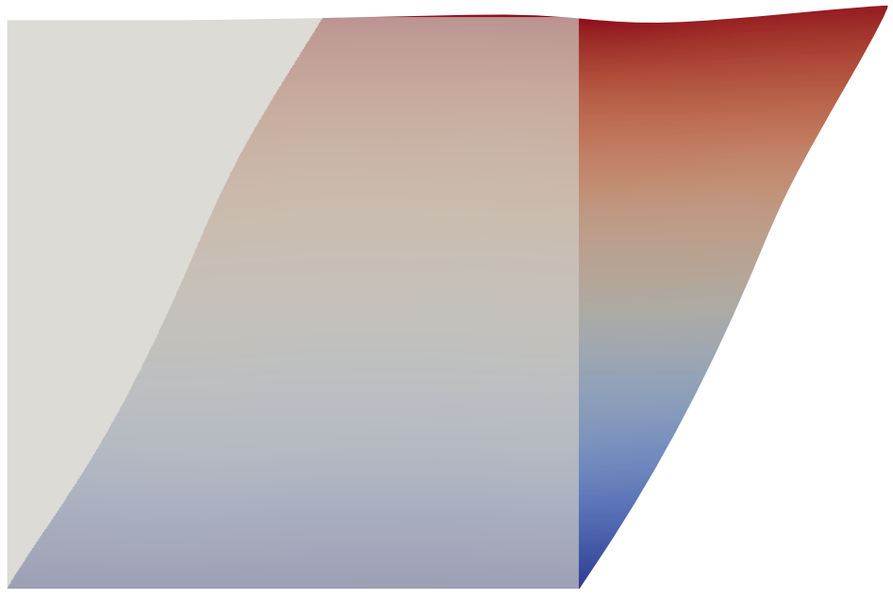
Overview

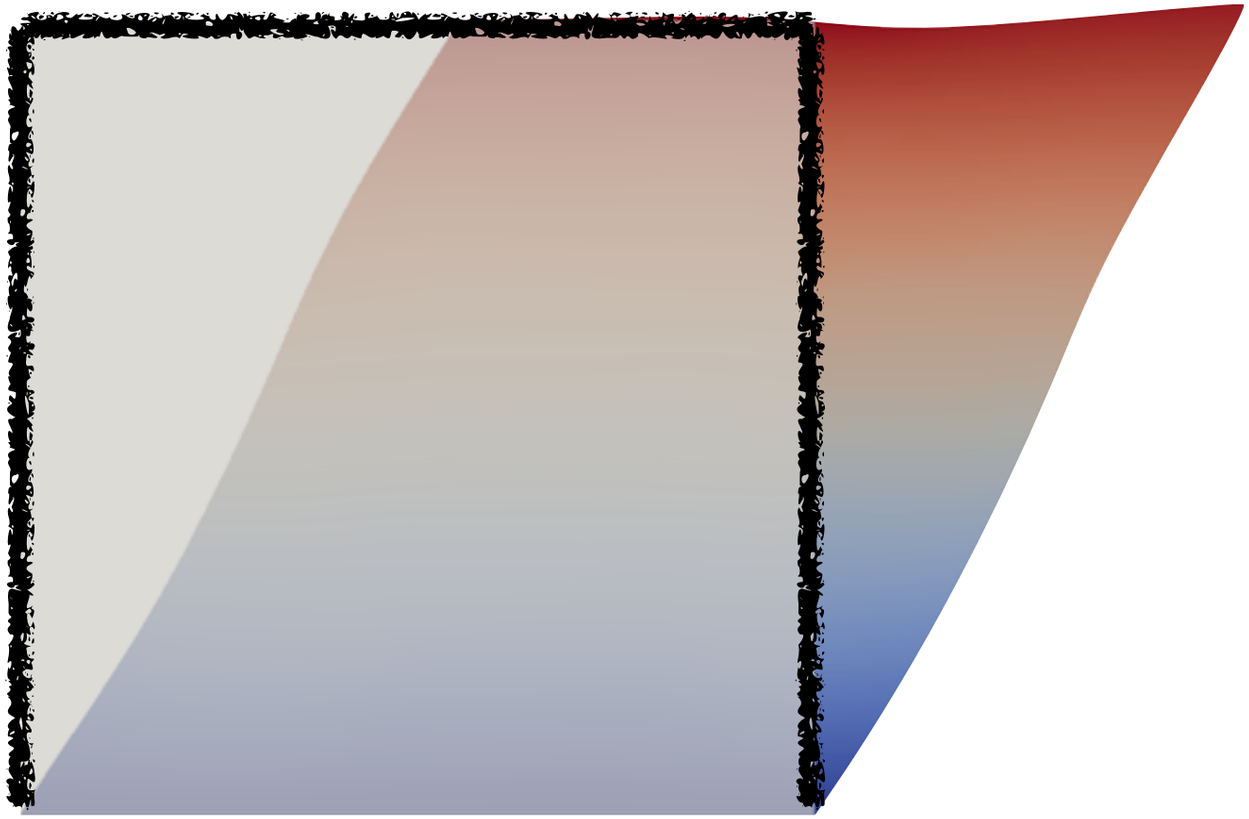
- Why?
- Bayesian approach to inversion.
- Relate classical optimisation techniques to the Bayesian inversion approach.
- Example problem: sparse surface observations of a solid block.
- Use dolfin-adjoint and petsc4py to solve the problem.
- Dealing with high-dimensional posterior covariance.

Why?


PHILIPS
L12-5/SmPrt AdBrst
FR 15Hz
R1
2D
77%
C 60
P Med
Res
ELASTO:
Opt 1
S Med
P High
D Med
M3 M2
A
E
P
Tx
4.0
BREAST CYST
ANECHOIC IMAGING

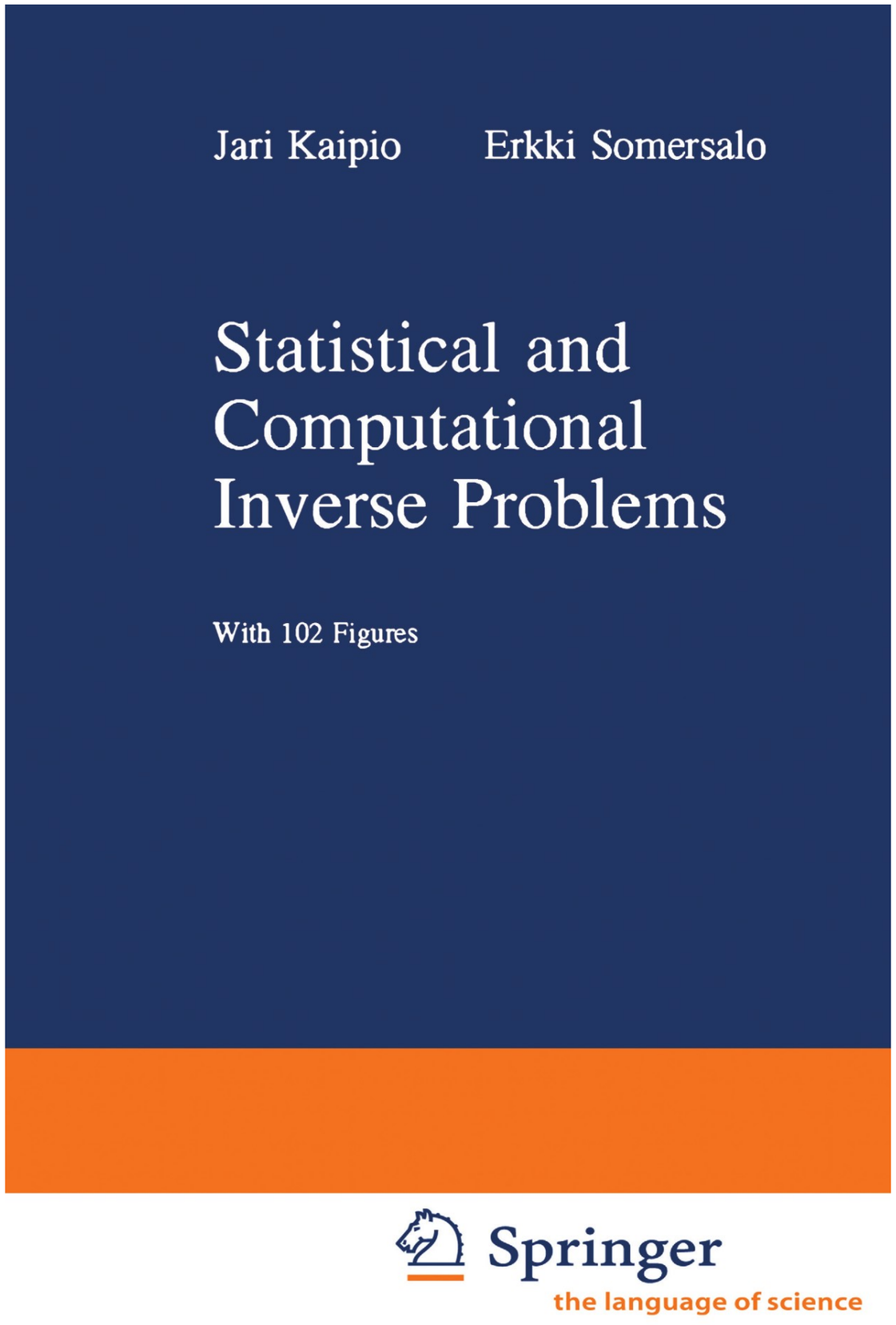

f_23
6.000e-01
0.5
0.4
3.000e-01

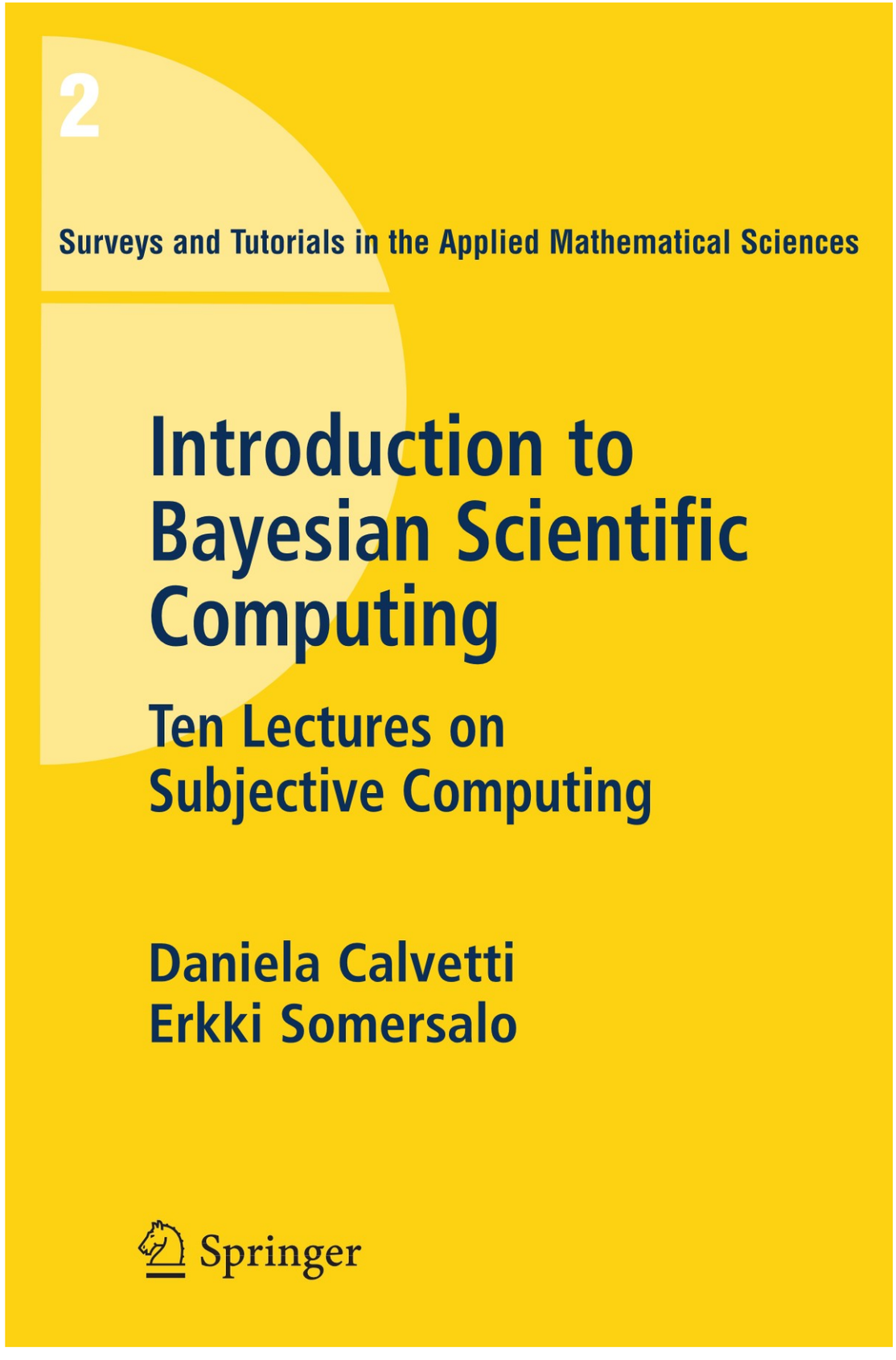




Q: What can we infer about the parameters inside the domain, just from displacement observations on the outside?

Q: Which parameters am I most uncertain about?


Jari Kaipio
Erkki Somersalo
Statistical and Computational Inverse Problems
With 102 Figures
Springer
the language of science


2
Surveys and Tutorials in the Applied Mathematical Sciences
Introduction to Bayesian Scientific Computing
Ten Lectures on Subjective Computing
Daniela Calvetti
Erkki Somersalo
Springer

Bayesian Approach

- *Deterministic event* - totally predictable.
- *Random event* - unpredictable.

- Bayesian approach to inverse problems:
  - The world is unpredictable.
  - Consider everything as a random variable.

Terminology

- Observation. *Displacements.* y
- Parameter. *Material property.* x
- Parameter-to-observable map. *Finite deformation hyperelasticity.* $f(x)$

Bayes Theorem

$$\pi_{\text{posterior}}(x \mid y) \propto \pi_{\text{likelihood}}(y \mid x)\pi_{\text{prior}}(x)$$

Goal: Given the observations, find the posterior distribution of the unknown parameters.

Three step plan

1. **Construct the prior.**
2. Construct the likelihood.
3. **Calculate/explore the posterior.**

Constructing a prior (with DOLFIN)

Must reflect our subjective belief about the unknown parameter.

Difficulty:

How to transfer qualitative information to quantitative.

Simple example involving a PDE solve: Smoothing Prior

https://bitbucket.org/snippets/jackhale/rk6xA

Reminder...

Let $x_0 \in \mathbb{R}^n$ and $\mathbf{\Gamma} \in \mathbb{R}^{n \times n}$ be a symmetric positive definite matrix. A multivariate Gaussian random variable X with mean x_0 and covariance $\mathbf{\Gamma}$ is a random variable with the probability density:

$$\pi(x) = \left(\frac{1}{2\pi|\Gamma|}\right) \exp\left(-\frac{1}{2}(x - x_0)^T \mathbf{\Gamma}^{-1}(x - x_0)\right).$$

When X follows a multivariate Gaussian, we use the following notation:

$$X \sim \mathcal{N}(x_0, \mathbf{\Gamma}).$$

Qualitative: *I think my parameter is smooth and is probably around zero at the boundary.*

Imagine a parameter related to a physical quantity in 1-dimensional space. Often, the value of parameter at a point is related to the value of the parameters next to it.

$$X_i = \frac{1}{2}(X_{i-1} + X_{i+1}) + W_j$$

With:

$$W = \mathcal{N}(0, \gamma^2 \mathbf{I})$$

$$\mathbf{A}X = W$$

```
mesh = UnitIntervalMesh(160)
V = FunctionSpace(mesh, “CG”, 1)
u = TrialFunction(V)
v = TestFunction(V)
...
a = (1.0/2.0)*h*inner(grad(u), grad(v))*dx
```

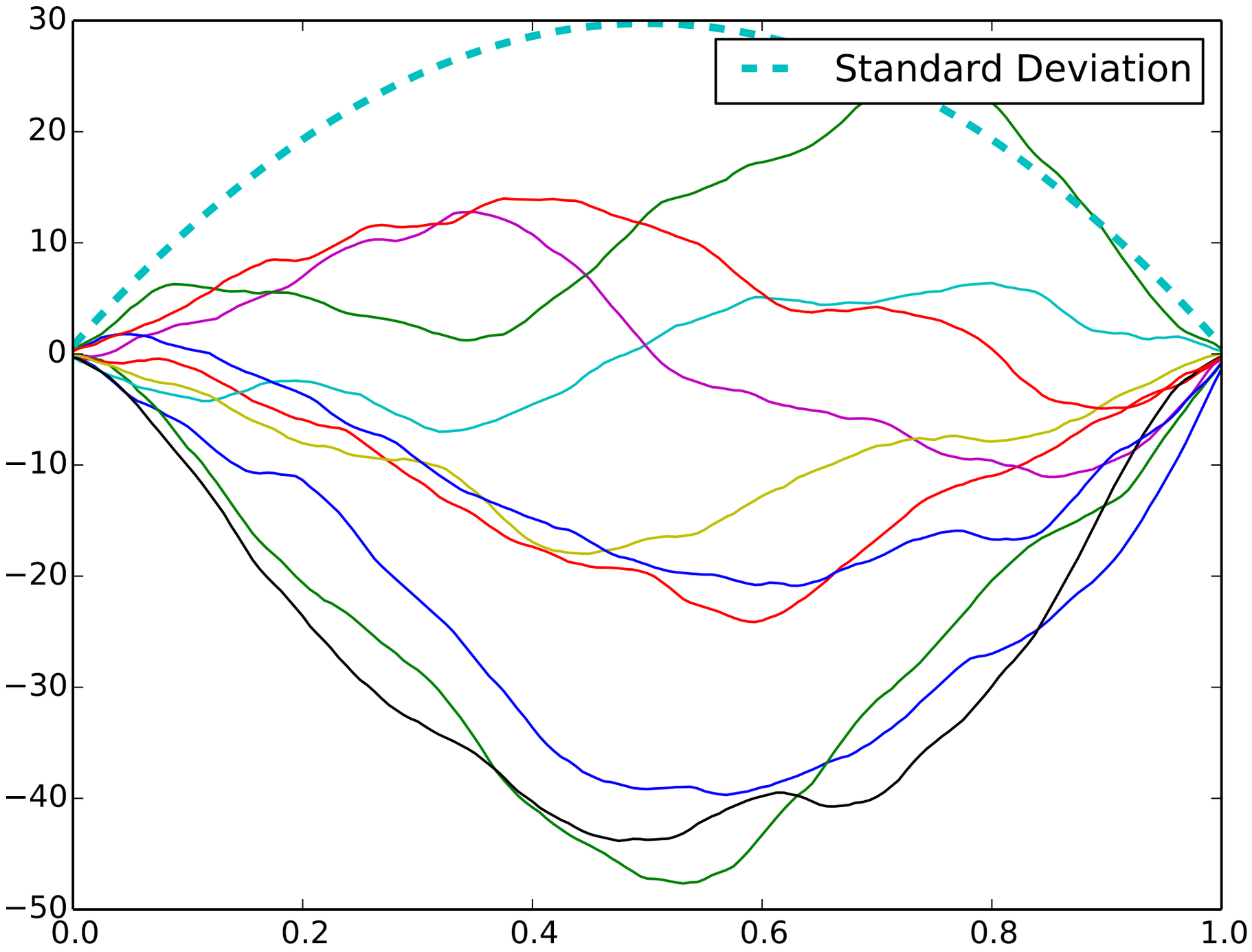
```
class W(Expression):
    def eval(self, value, x):
        value[0] = np.random.normal()
...
W = interpolate(W(), V)
A = assemble(A)
...
```

Boundary conditions

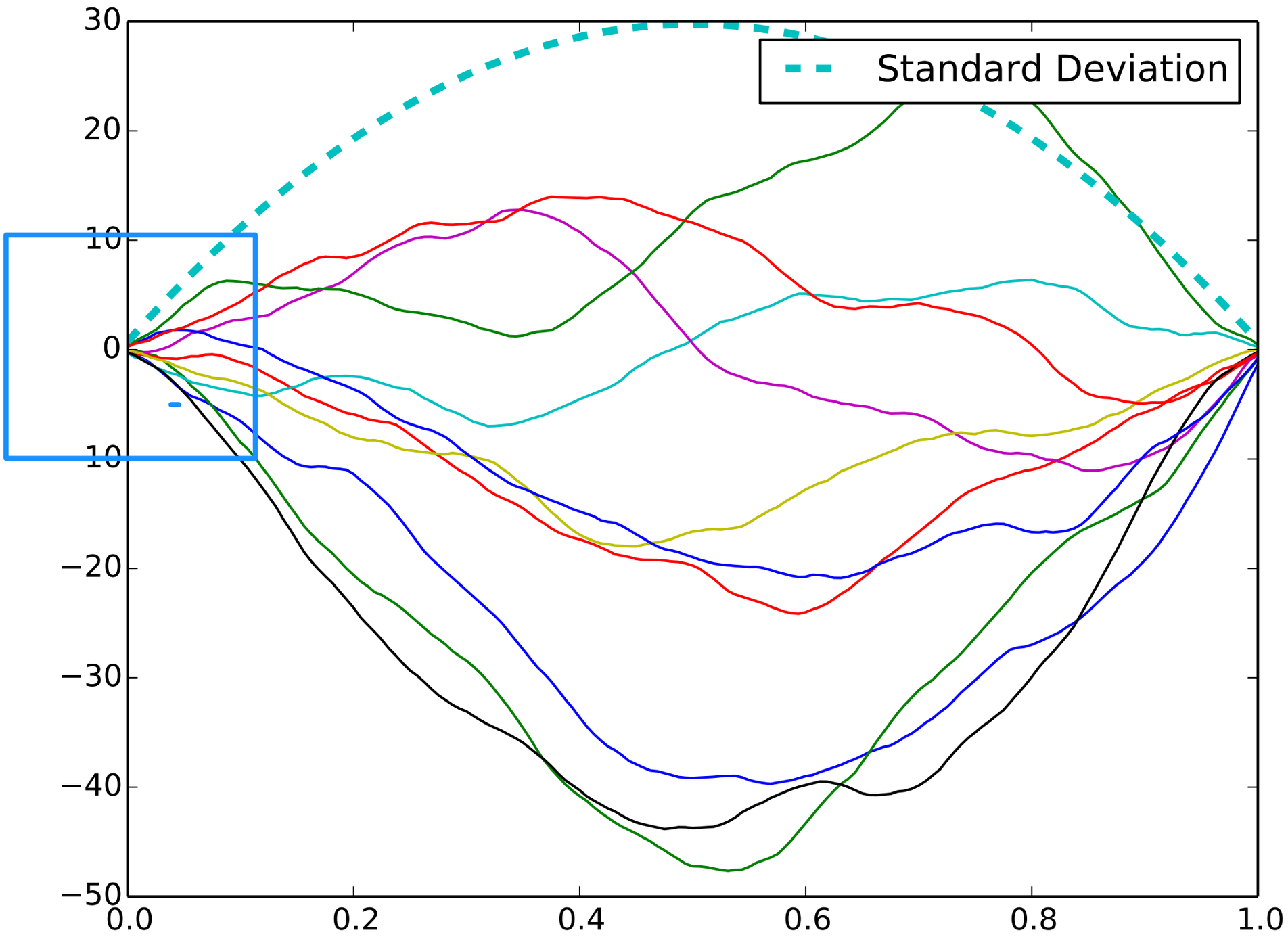
1. Dirichlet: set to zero.

2. Extend definition of Laplacian outside domain.

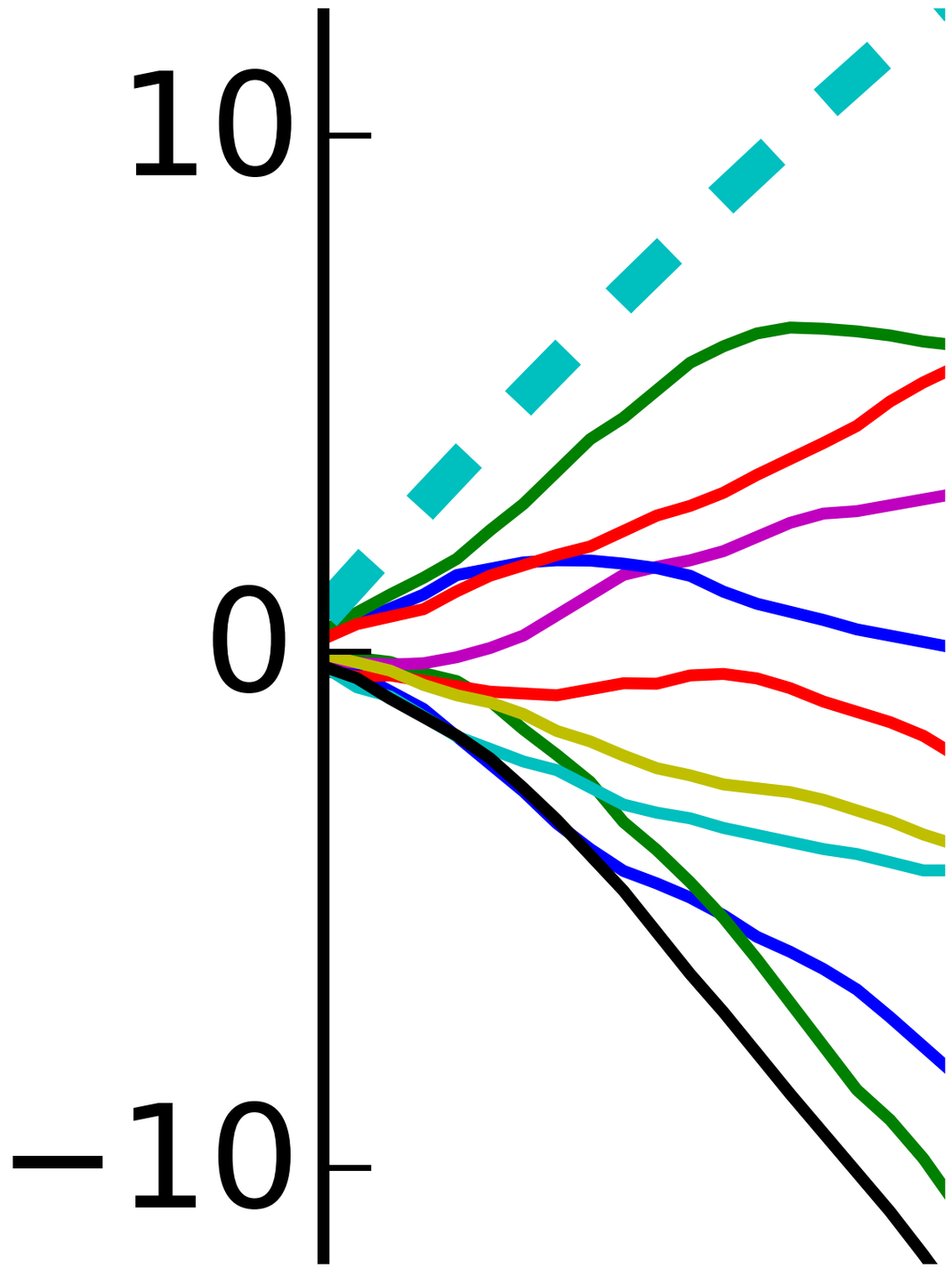
```
values = np.array([1.0, -0.5], dtype=np.float_)
rows = np.array([0], dtype=np.uintp)
cols = np.array([0, 1], \
       dtype=np.uintp) A.setrow(0, cols, values)
cols = np.array([V.dim() - 1, V.dim() - 2], \
                dtype=np.uintp)
A.setrow(V.dim() - 1, cols, values)
A.apply("insert")
```



$$\text{Std}(X, X) = \sqrt{\text{diag}(\gamma^2 \mathbf{A}^{-1} \mathbf{A}^{-T})}$$

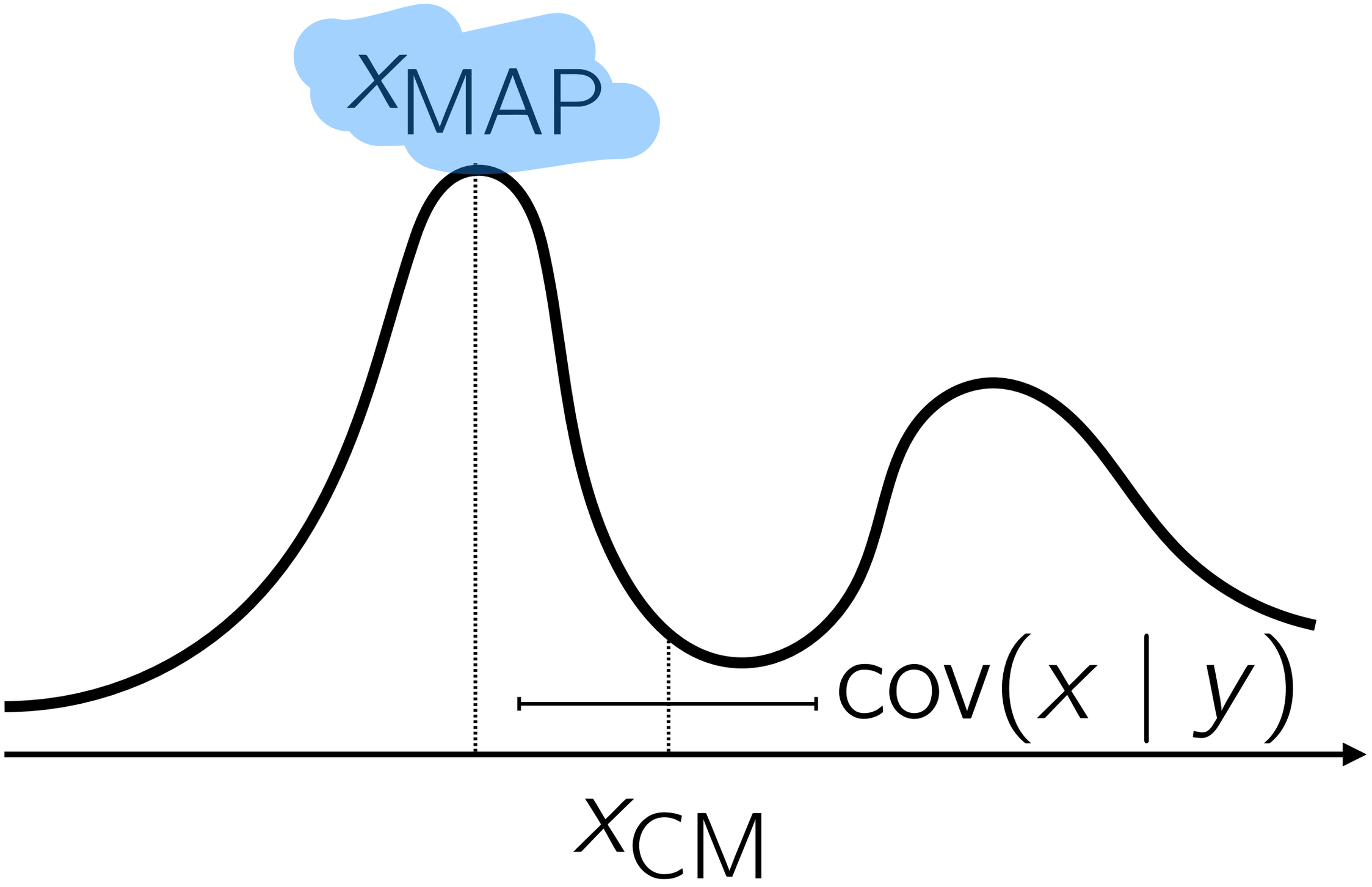


$$\mathrm{Std}(X, X) = \sqrt{\mathrm{diag}(\gamma^2 \mathbf{A}^{-1} \mathbf{A}^{-T})}$$

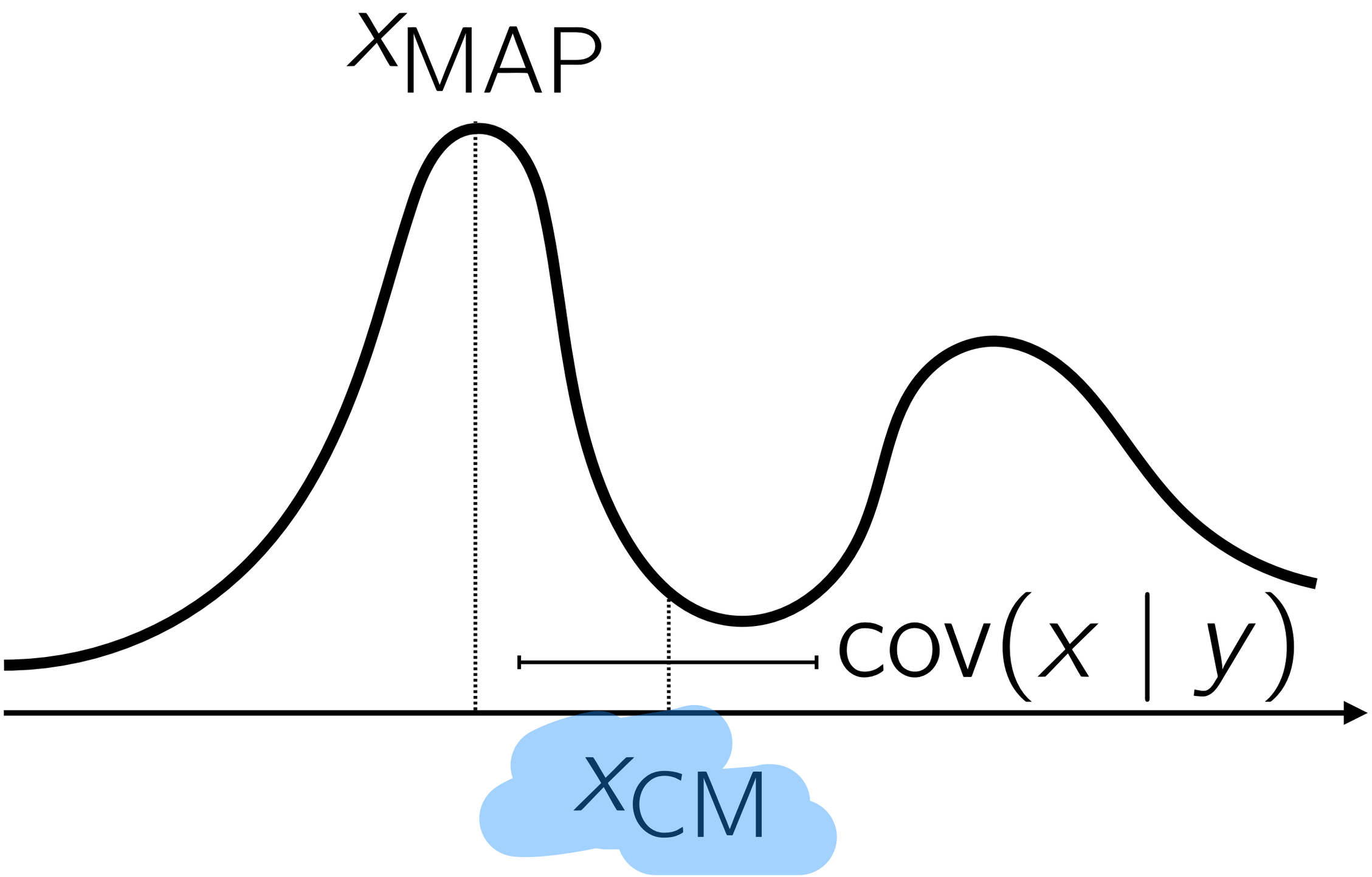

10
0
−10

Exploring the posterior

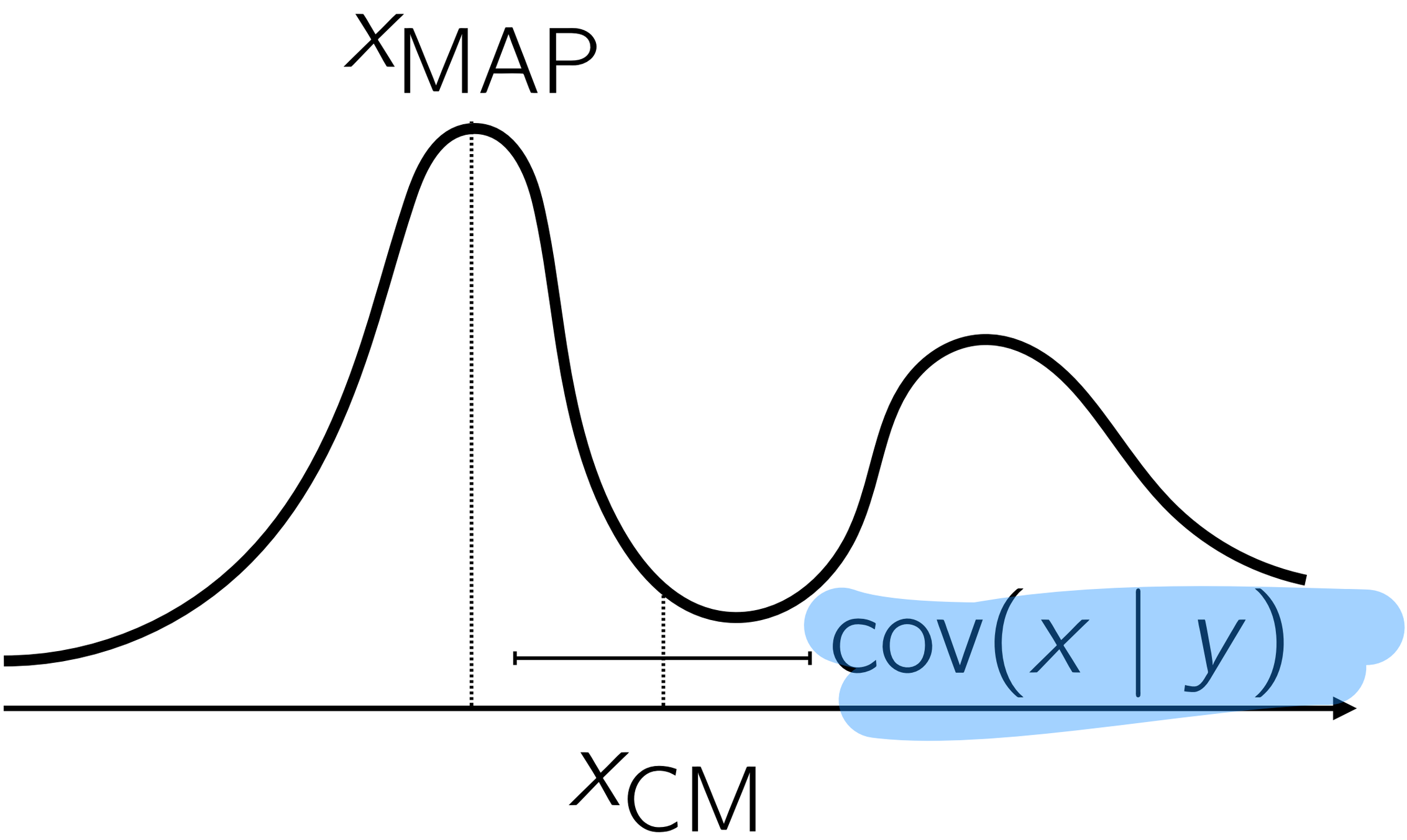
$$x_{\mathrm{MAP}} = \underset{x \in \mathbb{R}^n}{\arg\max}\, \pi_{\mathrm{posterior}}(x \mid y)$$



$$x_{\mathrm{CM}} = \int_{\mathbb{R}^n} x \pi_{\mathrm{posterior}}(x \mid y)\, dx$$



$$\mathrm{cov}(x \mid y) = \int_{\mathbb{R}^n} (x - x_{\mathrm{cm}})(x - x_{\mathrm{cm}})^T \pi_{\mathrm{posterior}}(x \mid y)\ dx \in \mathbb{R}^{n \times n}$$

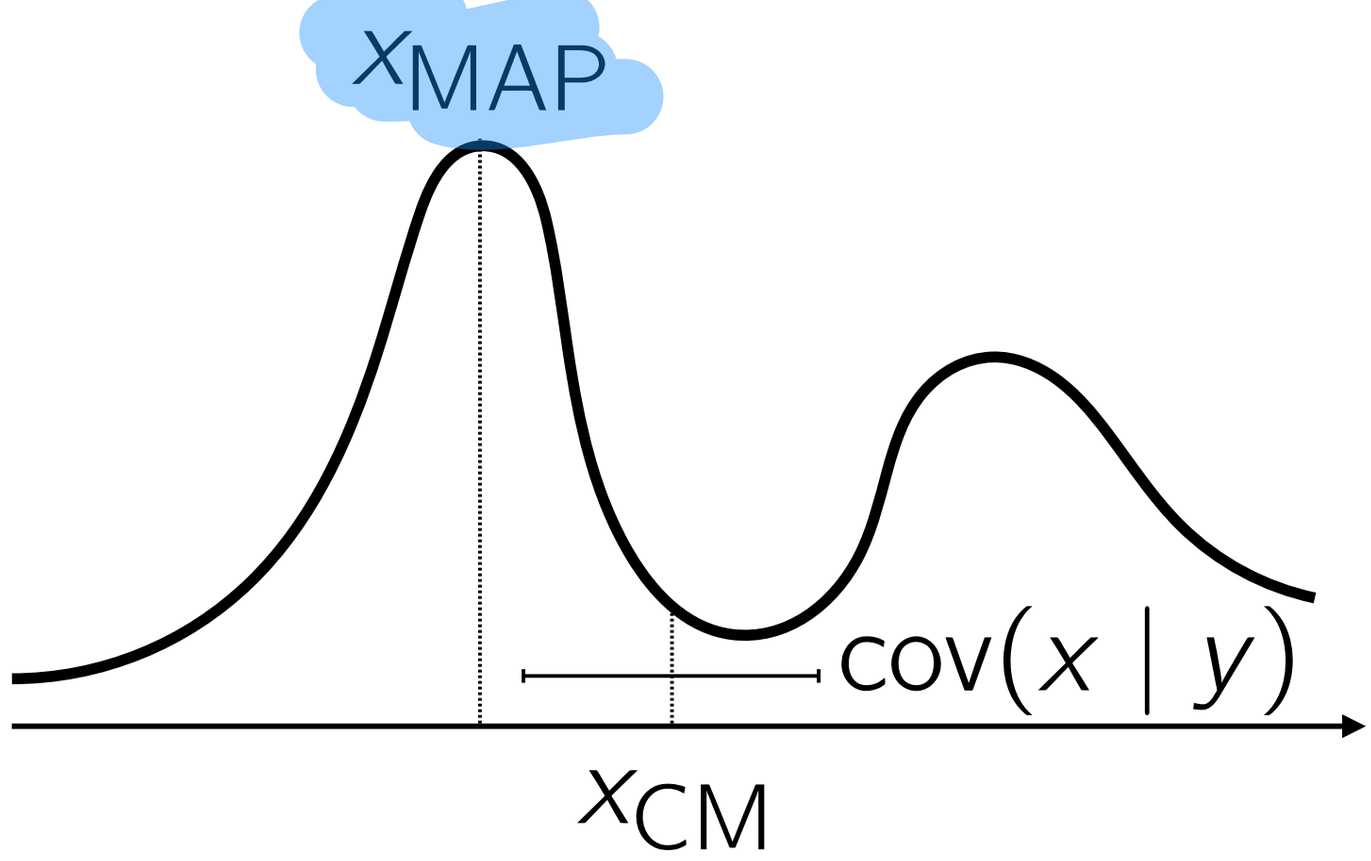


OK, but how can we use dolfin-adjoint to do this?

Aim: Connect *Bayesian approach* to *classical optimisation techniques.*

$$\pi_{\text{posterior}}(x \mid y) \propto \pi_{\text{likelihood}}(y \mid x)\pi_{\text{prior}}(x)$$

$$x_{\text{MAP}} = \underset{x \in \mathbb{R}^n}{\arg\max}\, \pi_{\text{posterior}}(x \mid y)$$



Assumptions

1. I think my parameter is Gaussian (prior).

2. My parameter to observable map is linear and my noise model is Gaussian.

$$X \sim \mathcal{N}(x_0, \mathbf{\Gamma}_{\text{prior}}), X \in \mathbb{R}^n$$

$$Y = AX + E, Y \in \mathbb{R}^m, A \in \mathbb{R}^{m \times n}$$

$$E \sim \mathcal{N}(0, \mathbf{\Gamma}_{\text{noise}}), Y \in \mathbb{R}^m$$

Plug it in…

$$\pi_{\text{posterior}}(x \mid y) \propto \pi_{\text{likelihood}}(y \mid x)\pi_{\text{prior}}(x)$$

$$\pi_{\text{posterior}}(x|y) \propto \exp\left(-\frac{1}{2}(y-\mathbf{A}x)^T\boldsymbol{\Gamma}_{\text{noise}}^{-1}(y-\mathbf{A}x)\right)\times\exp\left(-\frac{1}{2}(x-x_0)^T\boldsymbol{\Gamma}_{\text{prior}}^{-1}(x-x_0)\right)$$

$$\pi_{\text{posterior}}(x|y) \propto \exp\left(-\frac{1}{2}(y-\mathbf{A}x)^T\mathbf{\Gamma}_{\text{noise}}^{-1}(y-\mathbf{A}x)\right)\times\exp\left(-\frac{1}{2}(x-x_0)^T\mathbf{\Gamma}_{\text{prior}}^{-1}(x-x_0)\right)$$

$$x_{\text{MAP}} = \underset{x\in\mathbb{R}^n}{\arg\max}\,\pi_{\text{posterior}}(x \mid y)$$

$$x_{\text{MAP}} = \underset{x\in\mathbb{R}^n}{\arg\min}\left\{-\ln\pi_{\text{posterior}}(x \mid y)\right\}$$

$$\begin{aligned}-\ln\pi_{\text{posterior}}(x \mid y) &= \frac{1}{2}(y-\mathbf{A}x)^T\mathbf{\Gamma}_{\text{noise}}^{-1}(y-\mathbf{A}x) + \frac{1}{2}(x-x_0)^T\mathbf{\Gamma}_{\text{prior}}^{-1}(x-x_0)\\ &= \frac{1}{2}\|y-\mathbf{A}x\|^2_{\mathbf{\Gamma}_{\text{noise}}^{-1}} + \frac{1}{2}\|x-x_0\|^2_{\mathbf{\Gamma}_{\text{prior}}^{-1}}\end{aligned}$$

Optimise

$$
\begin{aligned}
g(x_{\mathsf{MAP}}) &:= \nabla_x \left(\frac{1}{2}\|y - \mathbf{A}x\|^2_{\mathbf{\Gamma}^{-1}_{\mathsf{noise}}} + \frac{1}{2}\|x - x_0\|^2_{\mathbf{\Gamma}^{-1}_{\mathsf{prior}}} \right) \Bigg|_{x=x_{\mathsf{map}}} \\
&= \mathbf{A}^T \mathbf{\Gamma}^{-1}_{\mathsf{noise}}(y - \mathbf{A}x_{\mathsf{map}}) + \mathbf{\Gamma}^{-1}_{\mathsf{prior}}(x_{\mathsf{map}} - x_0) \\
&= 0
\end{aligned}
$$

$$
x_{\mathsf{MAP}} = \left(\mathbf{\Gamma}^{-1}_{\mathsf{prior}} - \mathbf{A}^T \mathbf{\Gamma}^{-1}_{\mathsf{noise}} \mathbf{A} \right)^{-1} \left(\mathbf{A}^T \mathbf{\Gamma}_{\mathsf{noise}} y + \mathbf{\Gamma}_{\mathsf{prior}} x_0 \right)
$$

$$x_{\text{MAP}} = \left(\mathbf{\Gamma}_{\text{prior}}^{-1} - \mathbf{A}^T \mathbf{\Gamma}_{\text{noise}}^{-1} \mathbf{A}\right)^{-1} \left(\mathbf{A}^T \mathbf{\Gamma}_{\text{noise}} y + \mathbf{\Gamma}_{\text{prior}} x_0\right)$$

$$\mathbf{H} := \nabla_x g = \mathbf{\Gamma}_{\text{prior}}^{-1} - \mathbf{A}^T \mathbf{\Gamma}_{\text{noise}}^{-1} \mathbf{A}$$

$$
\begin{aligned}
-\ln \pi_{\text{posterior}}(x \mid y) &= \frac{1}{2}(y - \mathbf{A}x)^T \boldsymbol{\Gamma}_{\text{noise}}^{-1}(y - \mathbf{A}x) + \frac{1}{2}(x - x_0)^T \boldsymbol{\Gamma}_{\text{prior}}^{-1}(x - x_0) \\
&= \frac{1}{2}\|y - \mathbf{A}x\|^2_{\boldsymbol{\Gamma}_{\text{noise}}^{-1}} + \frac{1}{2}\|x - x_0\|^2_{\boldsymbol{\Gamma}_{\text{prior}}^{-1}}
\end{aligned}
$$

8.1.7 Sum of two squared forms

In vector formulation (assuming $\boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2$ are symmetric)

$$
-\frac{1}{2}(\mathbf{x} - \mathbf{m}_1)^T \boldsymbol{\Sigma}_1^{-1}(\mathbf{x} - \mathbf{m}_1) \tag{358}
$$

$$
-\frac{1}{2}(\mathbf{x} - \mathbf{m}_2)^T \boldsymbol{\Sigma}_2^{-1}(\mathbf{x} - \mathbf{m}_2) \tag{359}
$$

$$
= \quad -\frac{1}{2}(\mathbf{x} - \mathbf{m}_c)^T \boldsymbol{\Sigma}_c^{-1}(\mathbf{x} - \mathbf{m}_c) + C \tag{360}
$$

$$
\begin{aligned}
\boldsymbol{\Sigma}_c^{-1} &= \boldsymbol{\Sigma}_1^{-1} + \boldsymbol{\Sigma}_2^{-1} && (361)\\
\mathbf{m}_c &= (\boldsymbol{\Sigma}_1^{-1} + \boldsymbol{\Sigma}_2^{-1})^{-1}(\boldsymbol{\Sigma}_1^{-1}\mathbf{m}_1 + \boldsymbol{\Sigma}_2^{-1}\mathbf{m}_2) && (362)\\
C &= \frac{1}{2}(\mathbf{m}_1^T \boldsymbol{\Sigma}_1^{-1} + \mathbf{m}_2^T \boldsymbol{\Sigma}_2^{-1})(\boldsymbol{\Sigma}_1^{-1} + \boldsymbol{\Sigma}_2^{-1})^{-1}(\boldsymbol{\Sigma}_1^{-1}\mathbf{m}_1 + \boldsymbol{\Sigma}_2^{-1}\mathbf{m}_2) && (363)\\
&\quad -\frac{1}{2}\left(\mathbf{m}_1^T \boldsymbol{\Sigma}_1^{-1}\mathbf{m}_1 + \mathbf{m}_2^T \boldsymbol{\Sigma}_2^{-1}\mathbf{m}_2\right) && (364)
\end{aligned}
$$

$$
\begin{aligned}
-\ln \pi_{\text{posterior}}(x \mid y) &= \frac{1}{2}\|y - \mathbf{A}x\|^2_{\mathbf{\Gamma}^{-1}_{\text{noise}}} + \frac{1}{2}\|x - x_0\|^2_{\mathbf{\Gamma}^{-1}_{\text{prior}}} \\
&= \frac{1}{2}\|x - x_{\text{MAP}}\|_{\mathbf{H}}
\end{aligned}
$$

$$
\pi_{\text{posterior}} \sim \mathcal{N}(x_{\text{MAP}}, \mathbf{H}^{-1})
$$

Back to the problem…

Find x_{MAP} that satisfies:

$$\min_x \frac{1}{2}\|y - f(x)\|^2_{\mathbf{\Gamma}^{-1}_{\text{noise}}} + \frac{1}{2}\|x - x_0\|^2_{\mathbf{\Gamma}^{-1}_{\text{prior}}},$$

where the parameter-to-observable map $f : \mathbb{R}^n \to \mathbb{R}^m$ is is defined such that:

$$F(f(x), x) = D_v \int_\Omega \Pi(f(x), x)\, dx = 0 \quad \forall v \in H^1_D(\Omega),\, x \in L^2(\Omega),$$

where

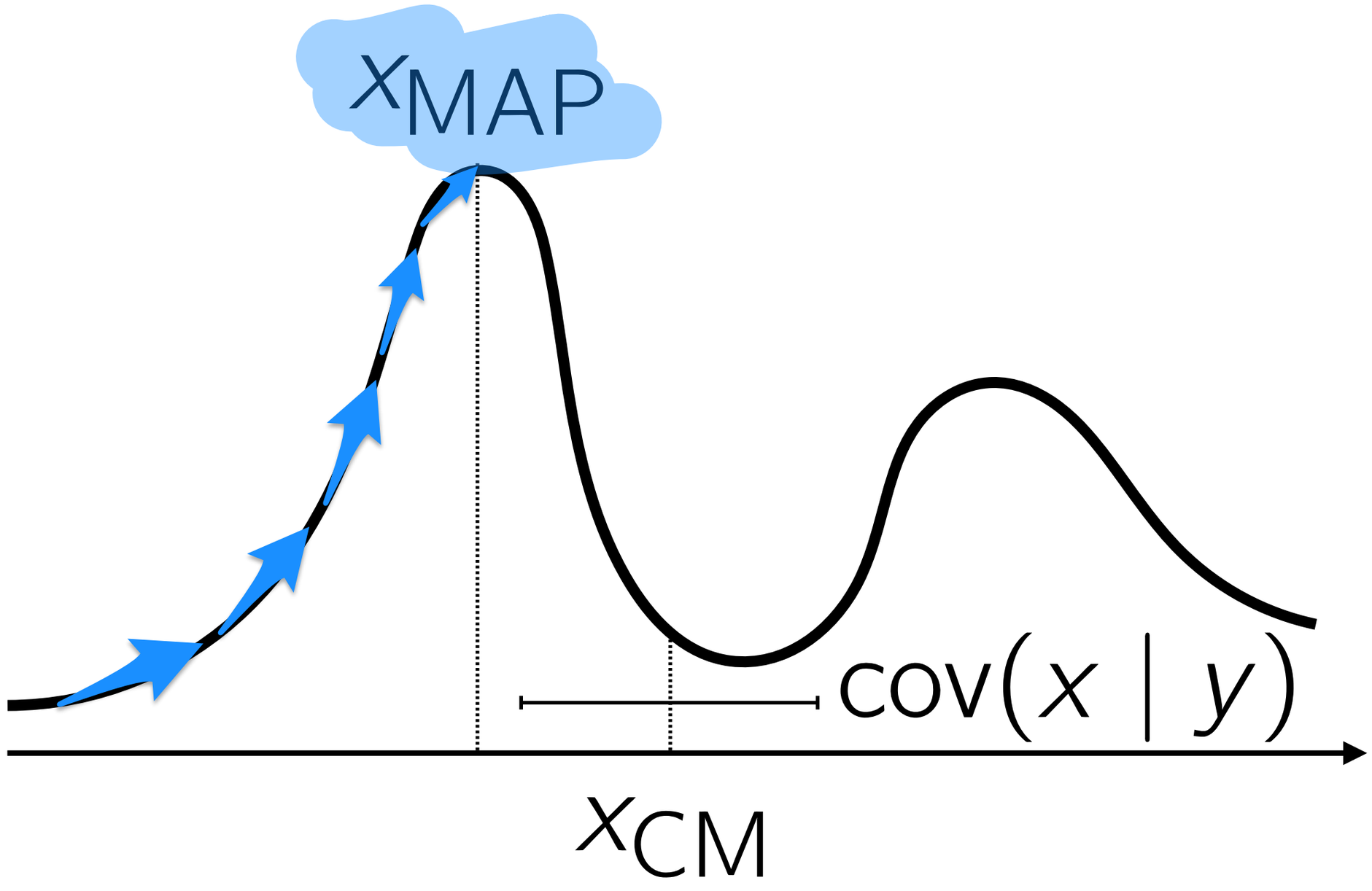
$$\begin{aligned}
\Pi(u, x) &= \int_\Omega \psi(u, x)\, dx - \int_\Gamma t \cdot u\, ds, \\
\psi(u, x) &= \frac{x}{2}(I_c - d) - x \ln(J) + \frac{\lambda}{2}\ln(J)^2, \\
\mathbf{F} &= \frac{\partial \phi}{\partial X} = \mathbf{I} + \nabla u, \\
\mathbf{C} &= \mathbf{F}^T\mathbf{F}, \\
\mathbf{I_C} &= \text{tr}(C), \\
\mathbf{J} &= \det\mathbf{F}.
\end{aligned}$$

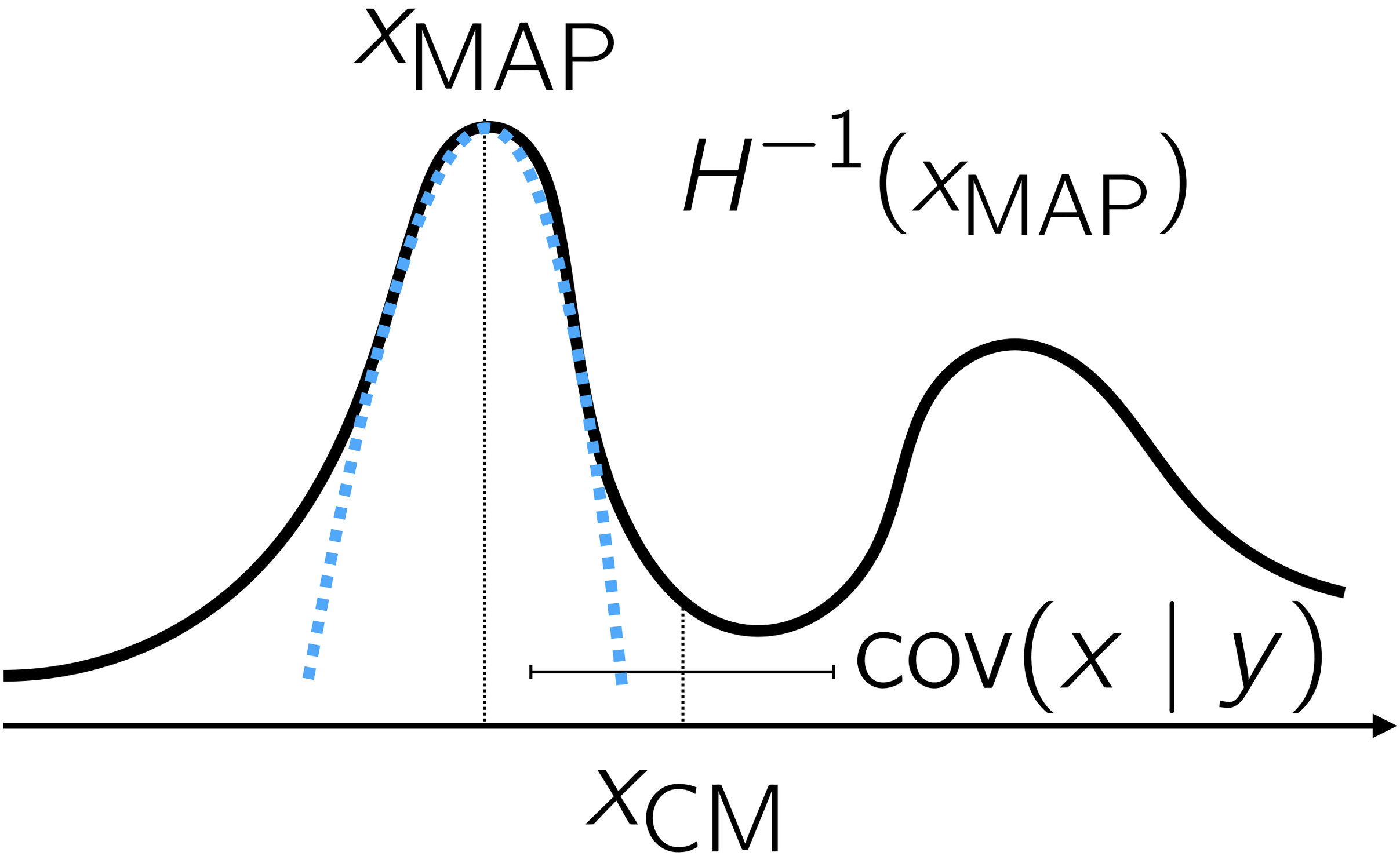
```
from dolfin import *
mesh = UnitSquareMesh(32, 32)

U = VectorFunctionSpace(mesh, "CG", 1)
V = FunctionSpace(mesh, "CG", 1)
# solution
u = Function(U)
# test functions
v = TestFunction(U)
# incremental solution
du = TrialFunction(U)
mu = interpolate(Constant(1.0), V)
lmbda = interpolate(Constant(100.0), V)

dims = mesh.type().dim()
I = Identity(dims)
F = I + grad(u)
C = F.T*F
J = det(F)
Ic = tr(C)

dims = mesh.type().dim()
I = Identity(dims)
F = I + grad(u)
C = F.T*F
J = det(F)
Ic = tr(C)
```

```
phi = (mu/2.0)*(Ic - dims) - mu*ln(J) + (lmbda/
2.0)*(ln(J))**2
Pi = phi*dx
# gateux derivative with respect to u in direction v
F = derivative(Pi, u, v)
# and with respect to u in direction du
J = derivative(F, u, du)

u_h = Function(U)
F_h = replace(F, {u: u_h})
J_h = replace(J, {u: u_h})
solve(F_h == 0, u_h, bcs, J=J_h)
```

```
u_obs << File("observations.xdmf")
J = Functional(inner(u - u_obs, u - u_obs)*dx + \
               inner(mu, mu))
m = Control(mu)
J_hat = ReducedFunctional(J, m)
...
dJdm = Jhat.derivative()[0]
H = Jhat.hessian(dm)[0]
```

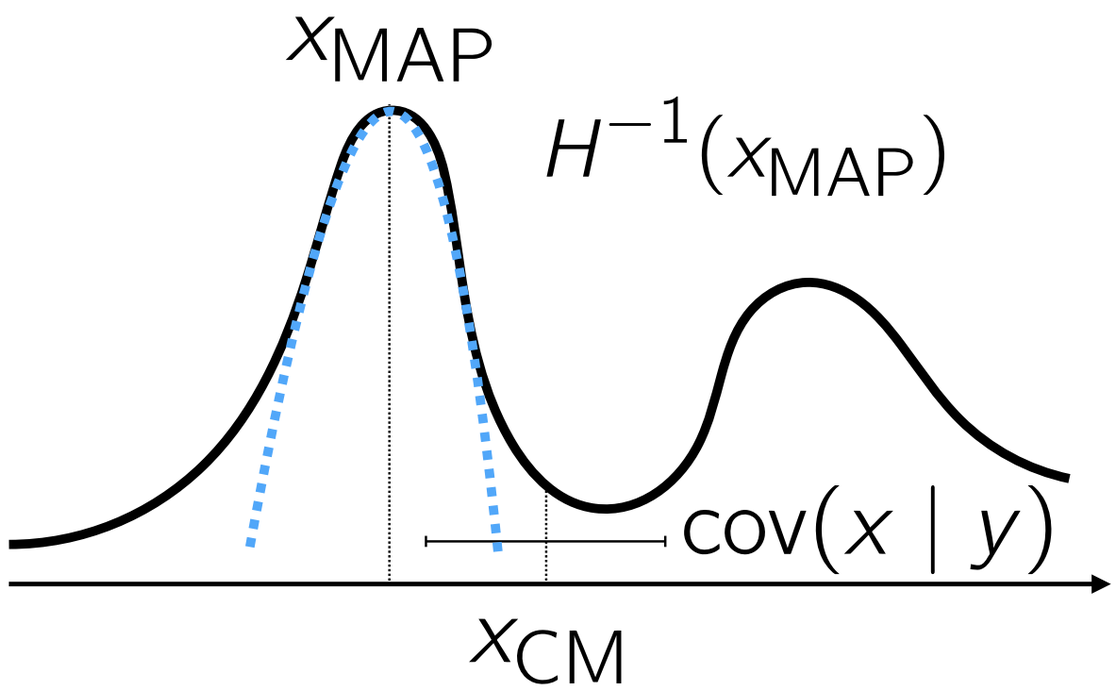
Wait a second…




x_{MAP}
$H^{-1}(x_{MAP})$
$\mathrm{cov}(x \mid y)$
x_{CM}

$$\pi_{\text{posterior}} \sim \mathcal{N}(x_{\text{MAP}}, \mathbf{H}^{-1})$$

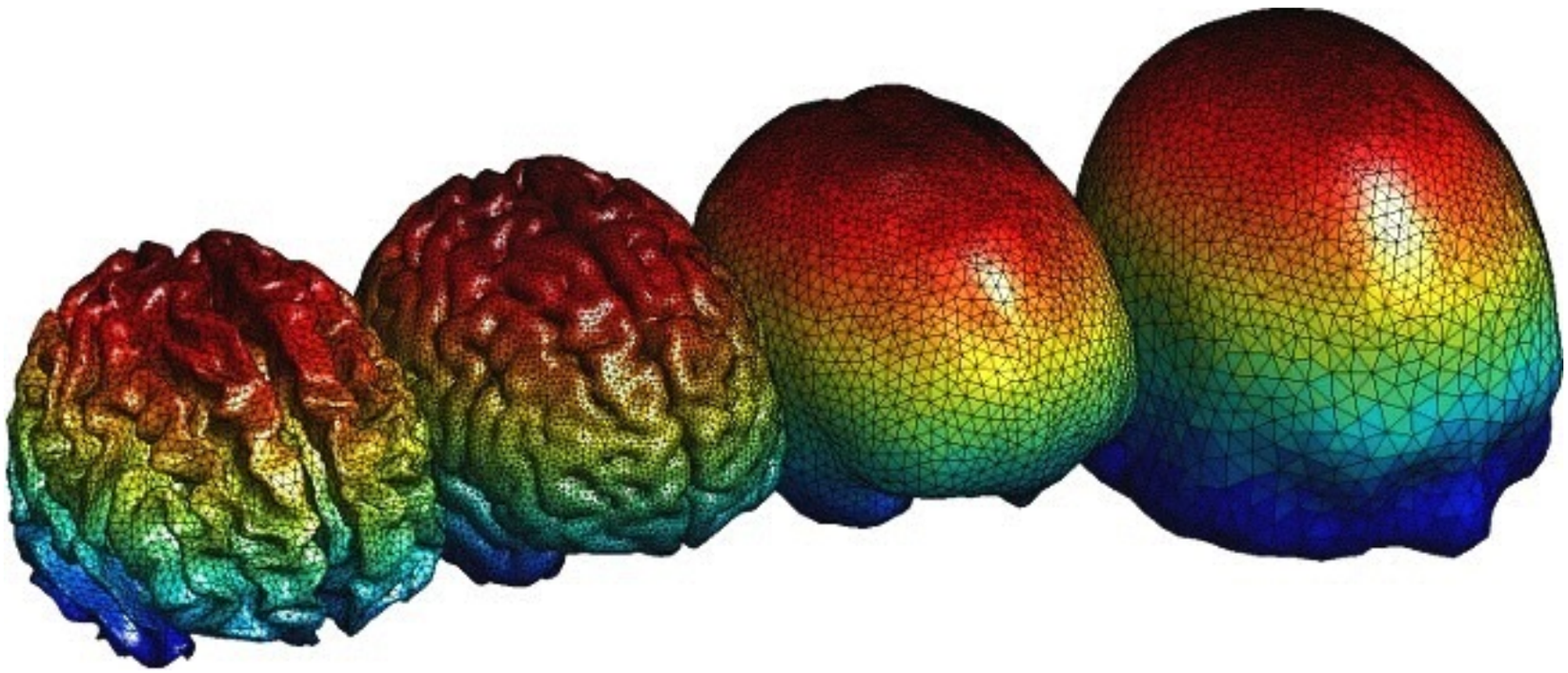
$$\pi^{\text{approx}}_{\text{posterior}} \sim \mathcal{N}(x_{\text{MAP}}, \mathbf{H}^{-1}(x_{\text{MAP}}))$$



Solving

Strategy: Use hooks in dolfin-adjoint to solve with petsc4py-based contexts.

- Parameter-to-observable map. Newton-Krylov method.
    - Inner solve with GAMG preconditioned GMRES (KSP) with near-null-space set.
    - Newton with second-order backtracking line search (SNES).

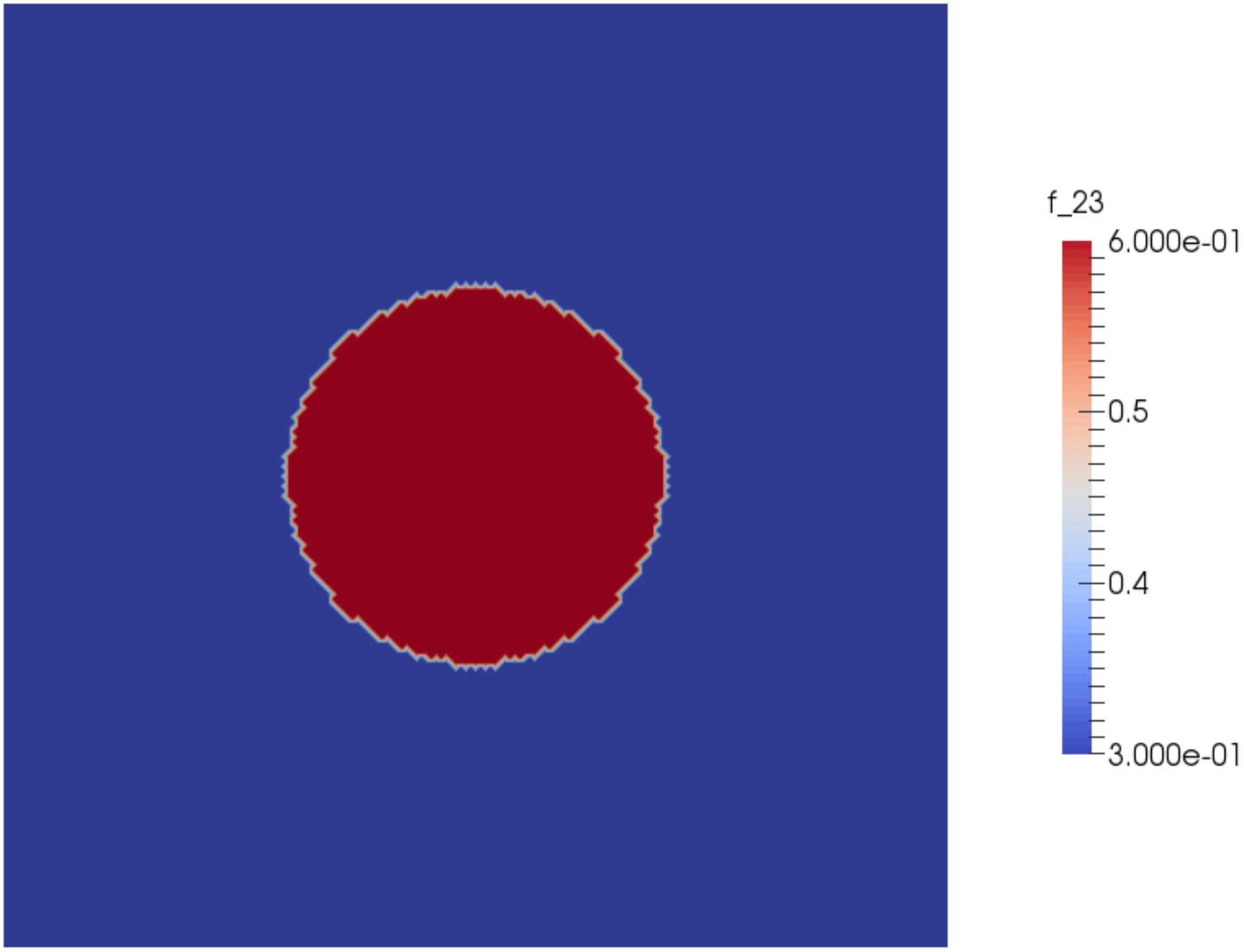


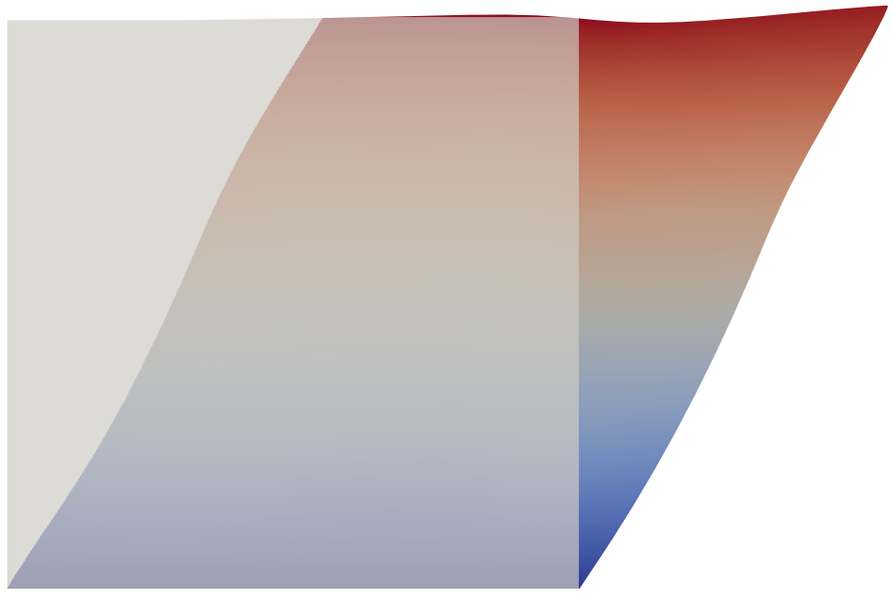
Colin27 brain atlas

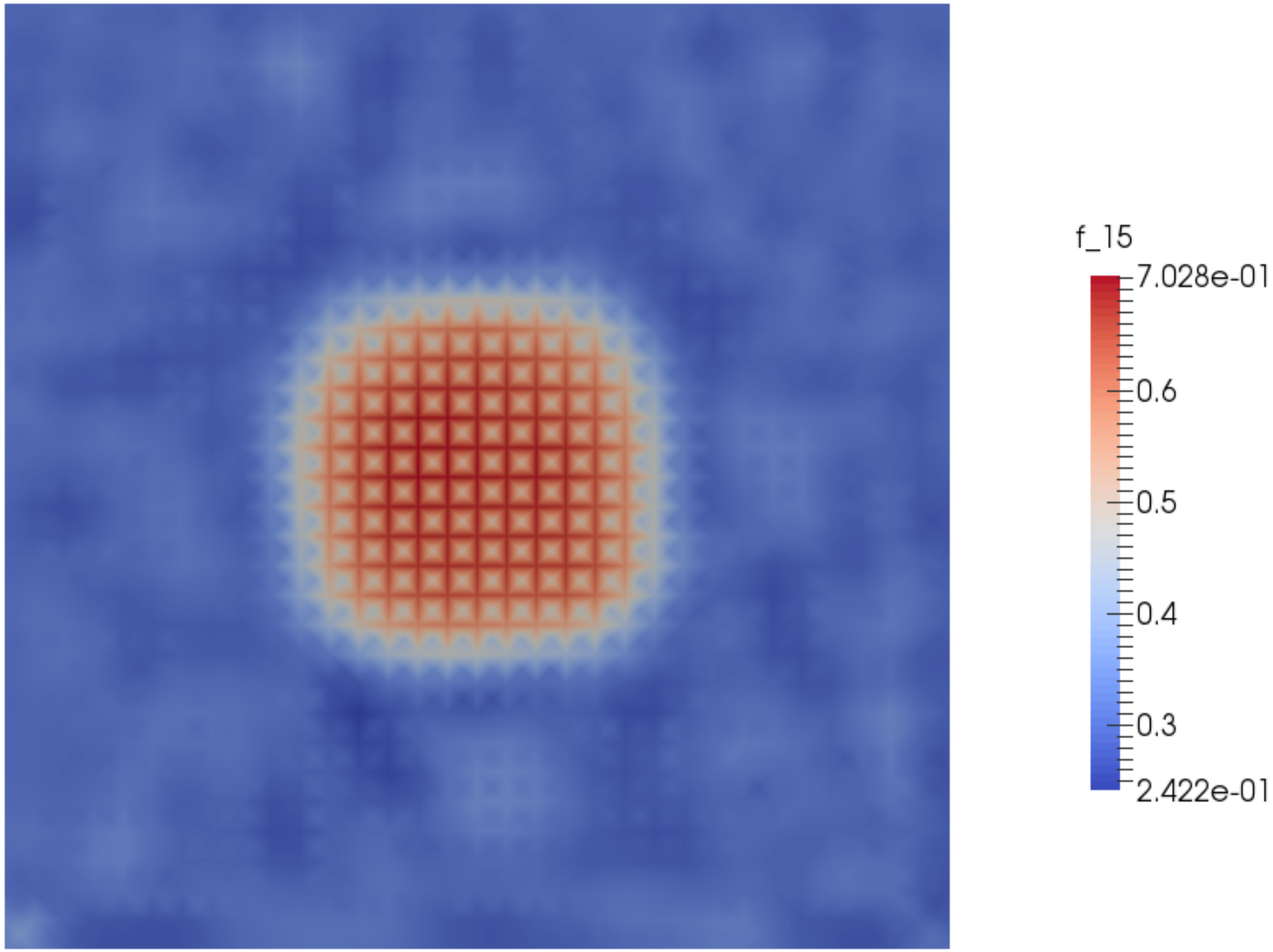
- 20% extension test, 16 Core Xeon, 1.12 million cells, ~29 secs to residual of 1E-10.

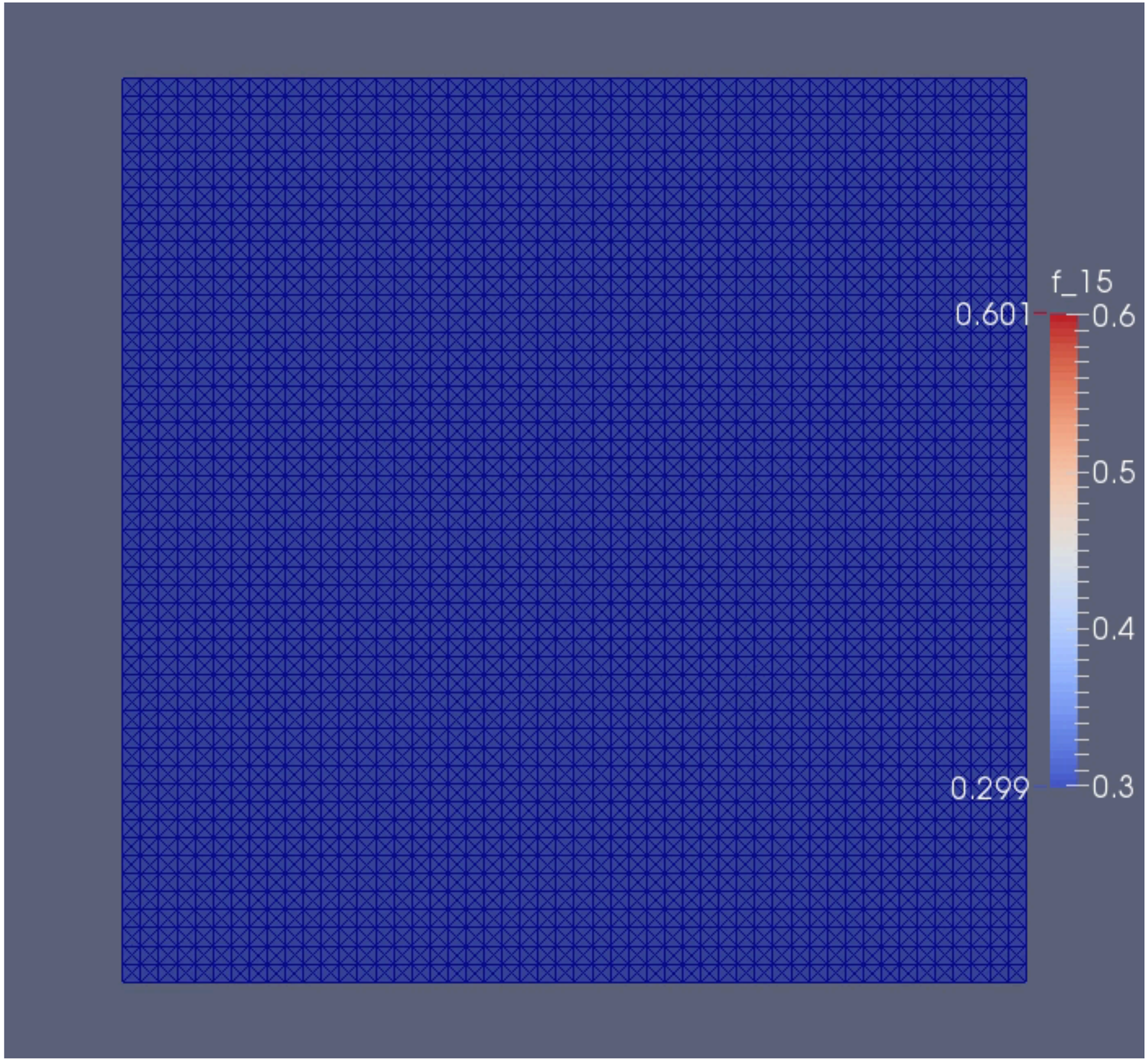
- MAP estimator.
  - Bound constrained Quasi-Newton BLMVM with More-Thuente line search (TAO).
- No Riesz map.

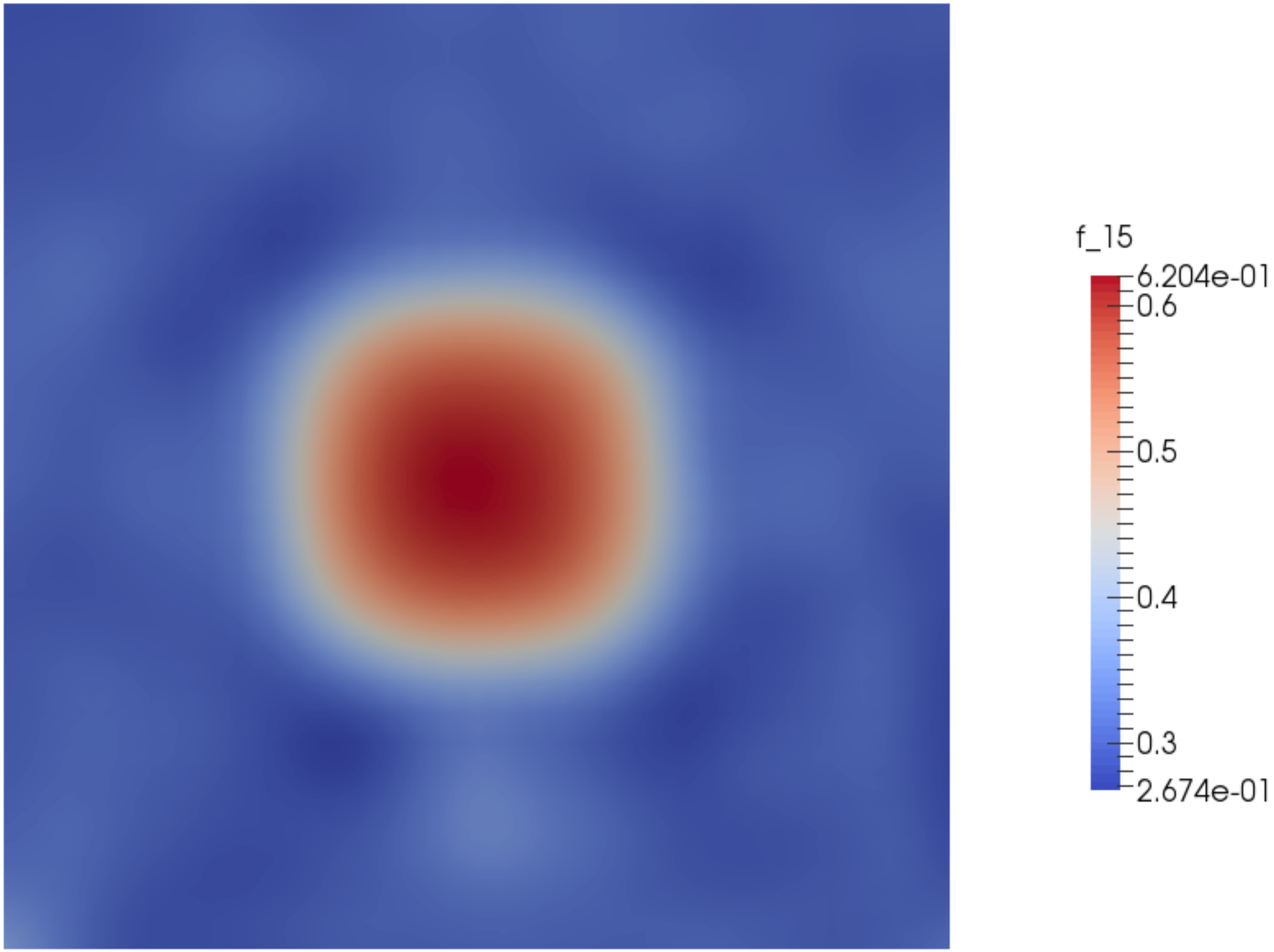
- Principal Component Analysis.
  - Trailing: BLOPEX Locally Optimal Block Preconditioned Gradient method (SLEPc/ BLOPEX).
  - Leading: Krylov Schur (SLEPc).


f_23
6.000e-01
0.5
0.4
3.000e-01

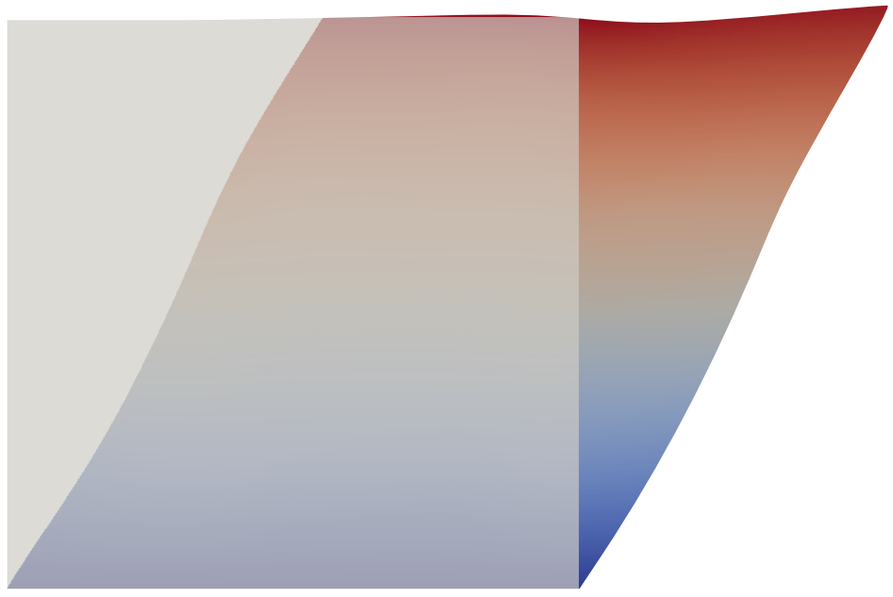


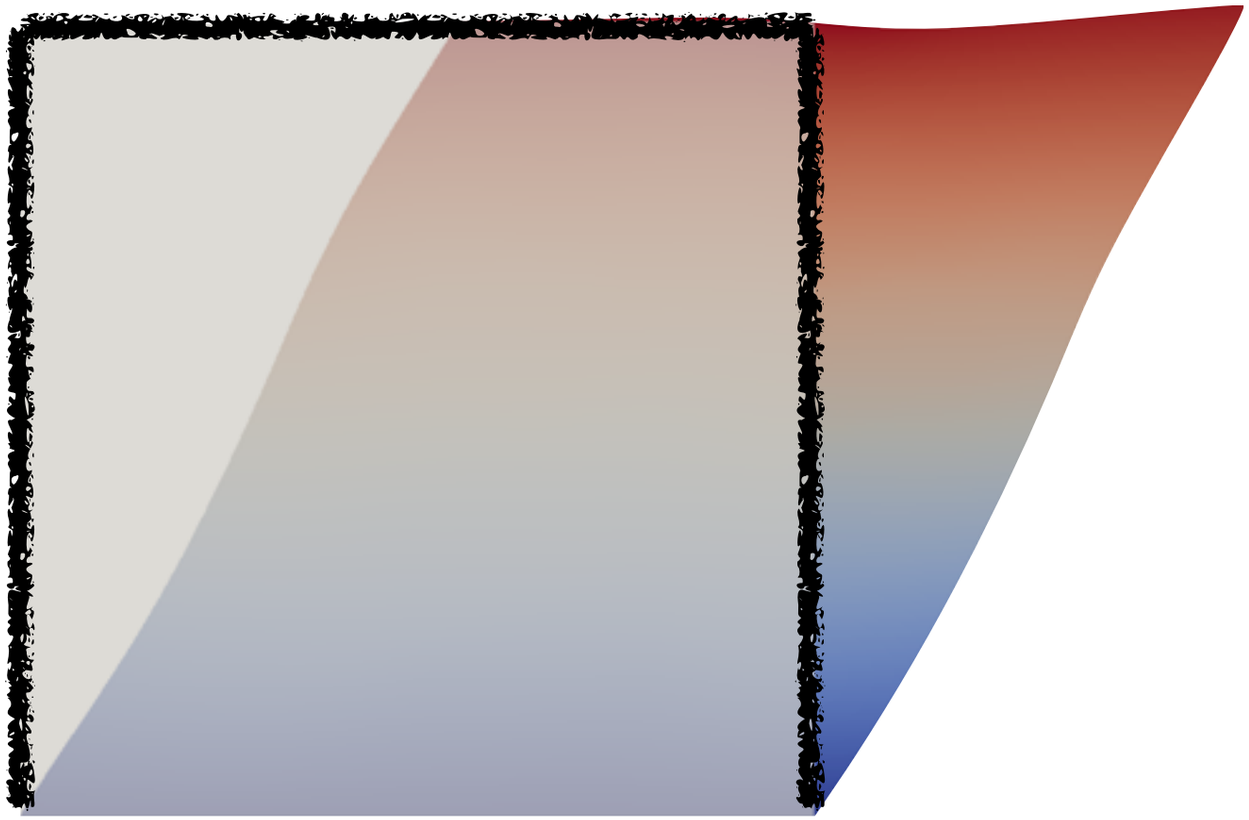

f_15
7.028e-01
0.6
0.5
0.4
0.3
2.422e-01


f_15
0.601
0.6
0.5
0.4
0.299
0.3


f_15
6.204e-01
0.6
0.5
0.4
0.3
2.674e-01

Back to uncertainty quantification…



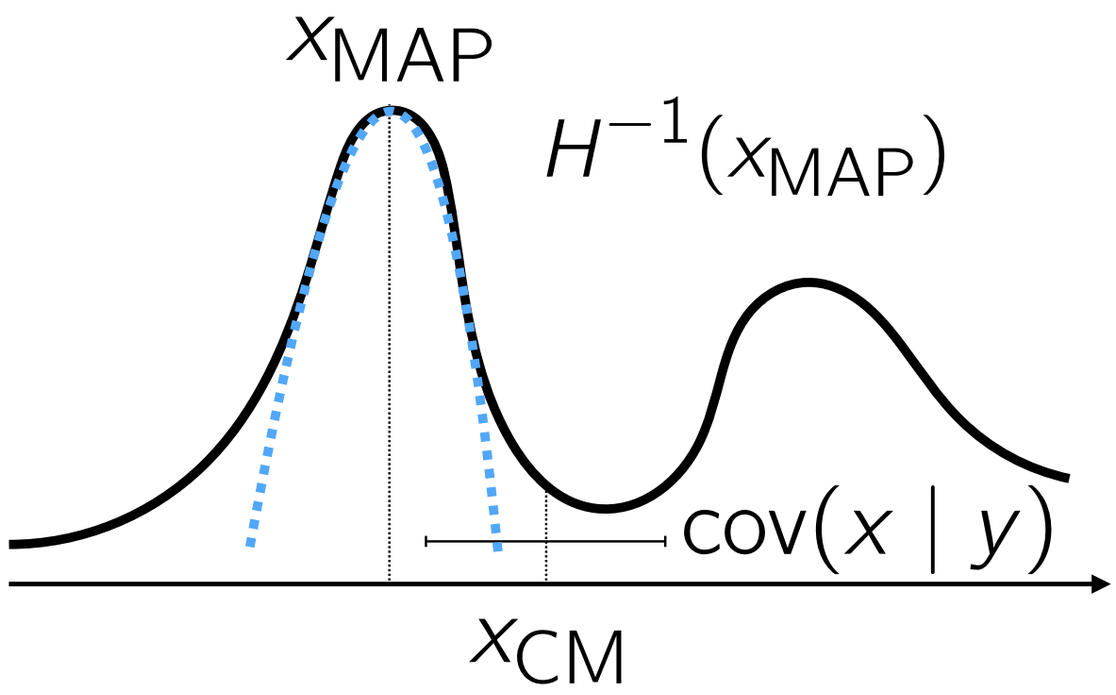


Q: What can we infer about the parameters inside the domain, just from displacement observations on the outside?

Q: Which parameters am I most uncertain about?

$$\pi_{\text{posterior}} \sim \mathcal{N}(x_{\text{MAP}}, \mathbf{H}^{-1})$$

$$\pi^{\text{approx}}_{\text{posterior}} \sim \mathcal{N}(x_{\text{MAP}}, \mathbf{H}^{-1}(x_{\text{MAP}}))$$



Big

Dense

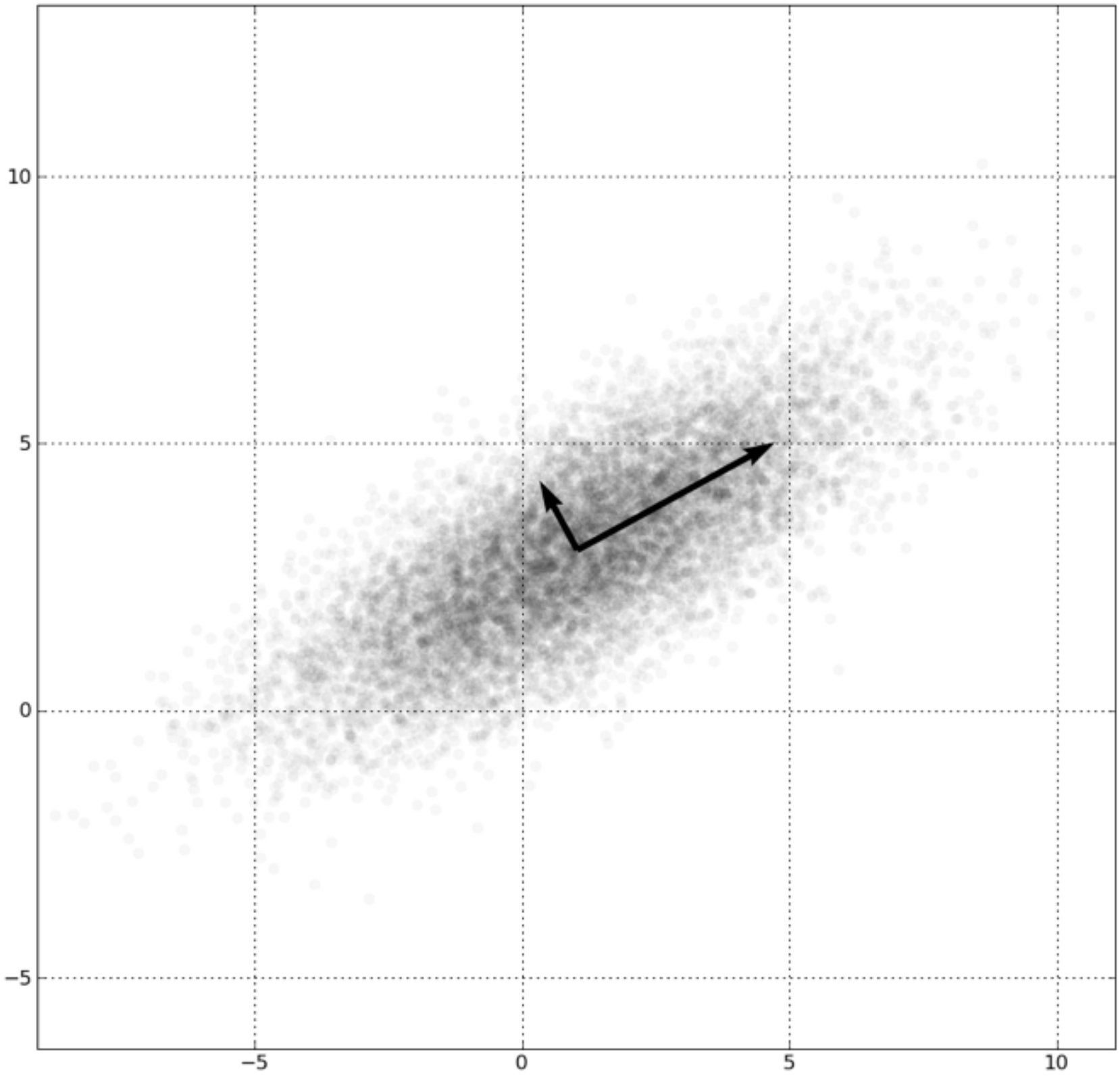
$$\mathbf{H} \in \mathbb{R}^{n \times n}$$

Expensive to calculate

Only have action

$$\mathbf{H} = \mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^T$$

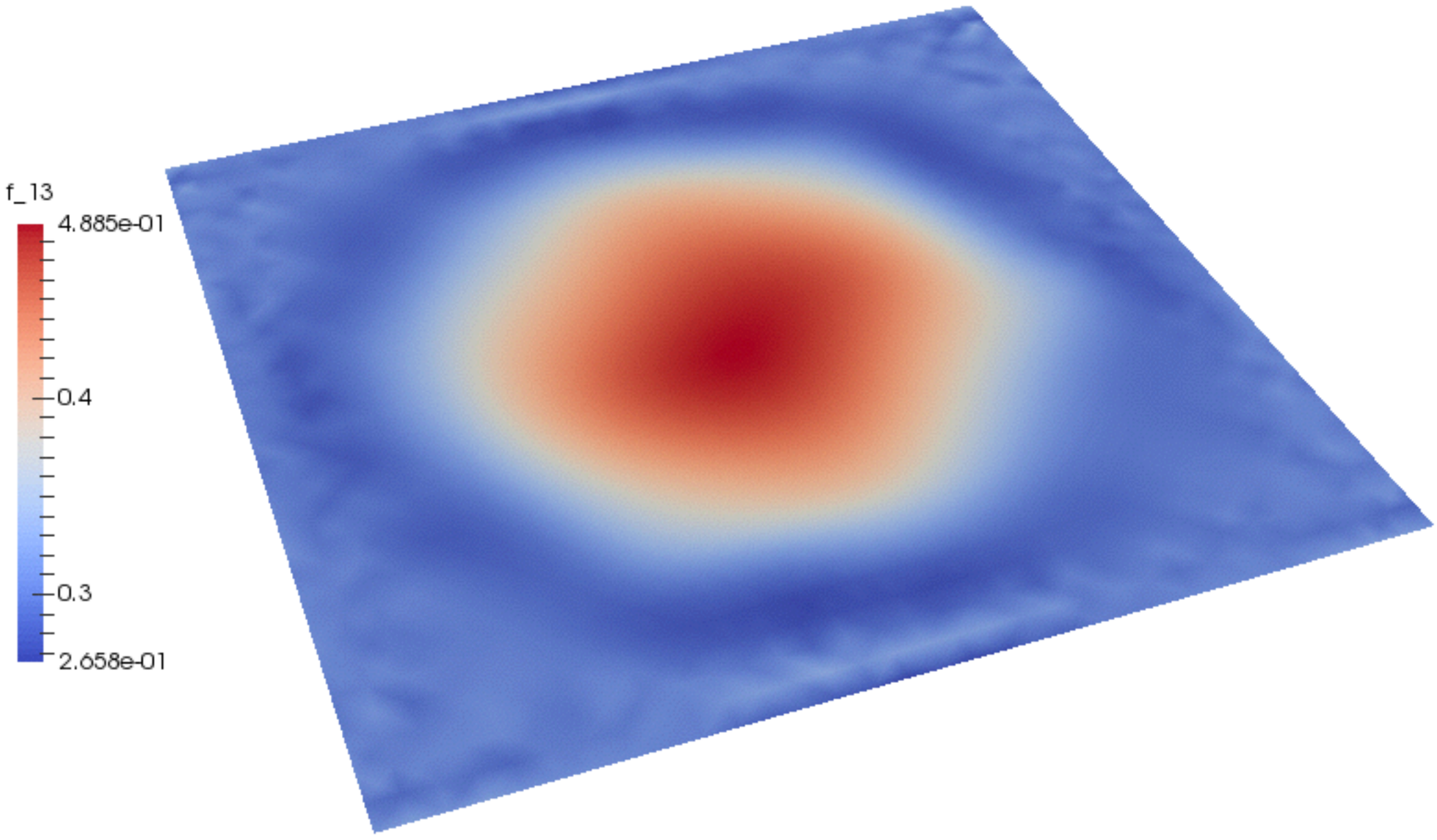
$$\boldsymbol{\Gamma}_{\text{post}} = \mathbf{H}^{-1} = \mathbf{Q}^T\boldsymbol{\Lambda}^{-1}\mathbf{Q}$$



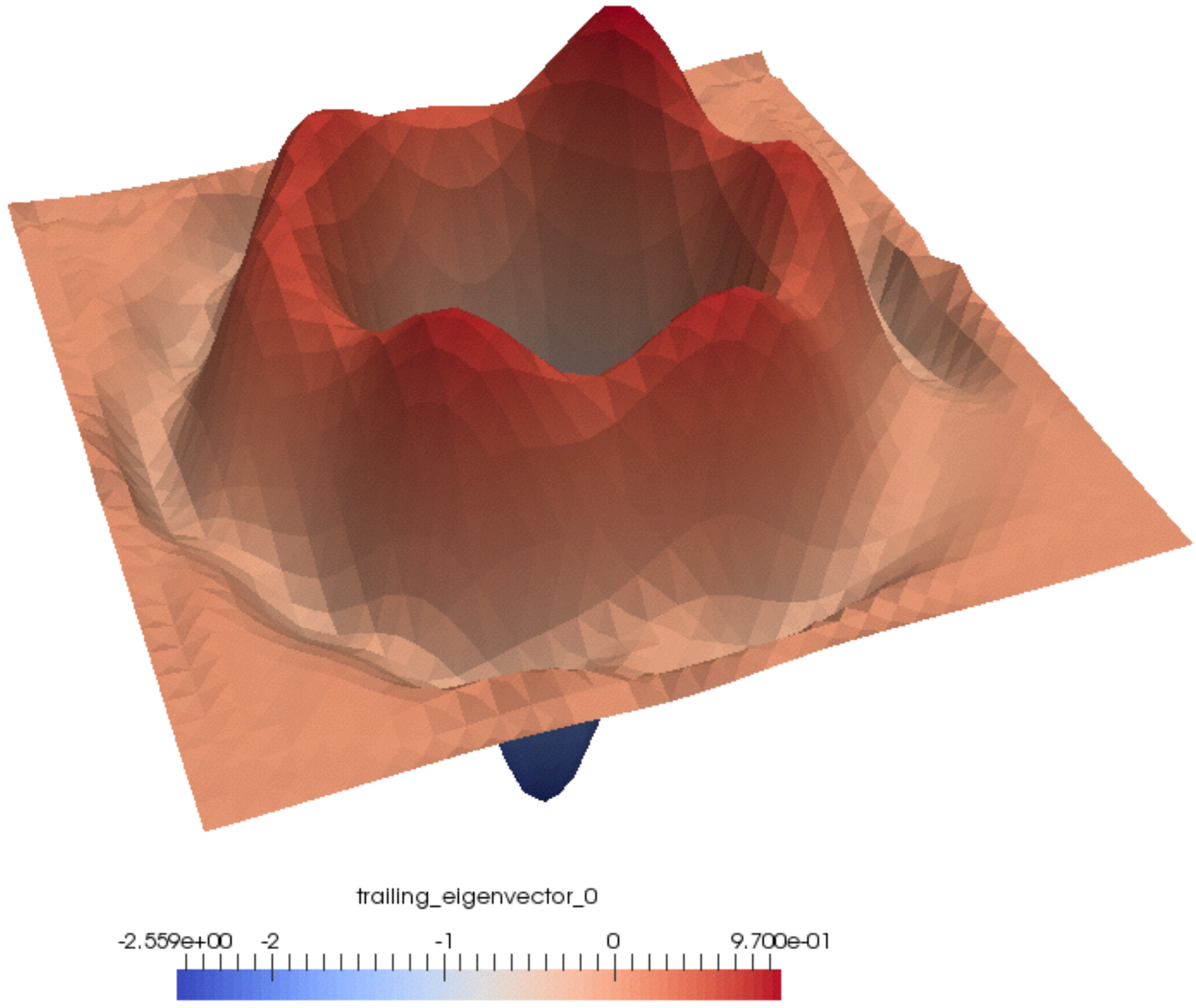
Wikipedia Commons

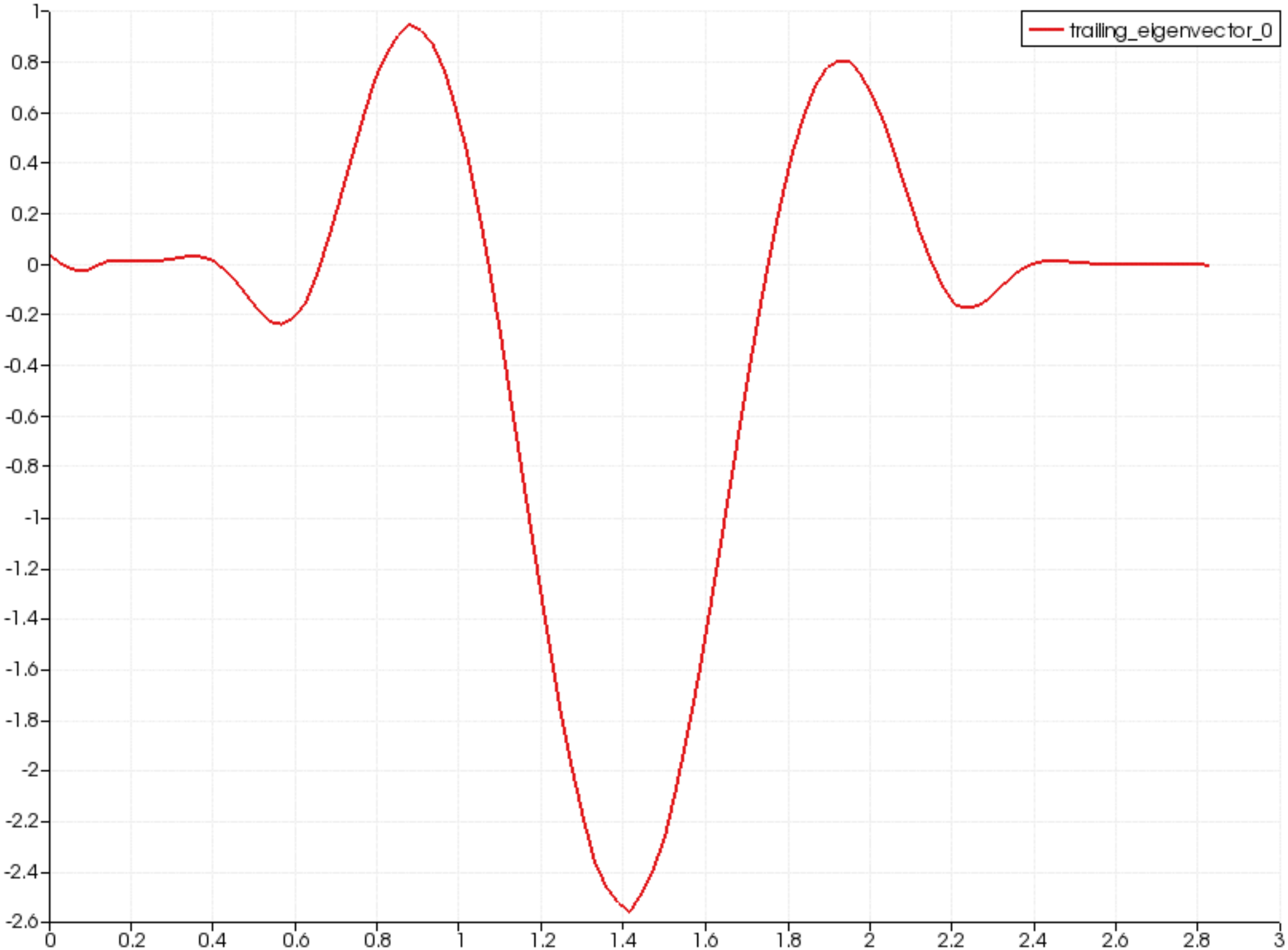
Trailing Eigenvector

Direction in parameter space *least* constrained by the observations



X_{map}


trailing_eigenvector_0
-2.559e+00
-2
-1
0
9.700e-01

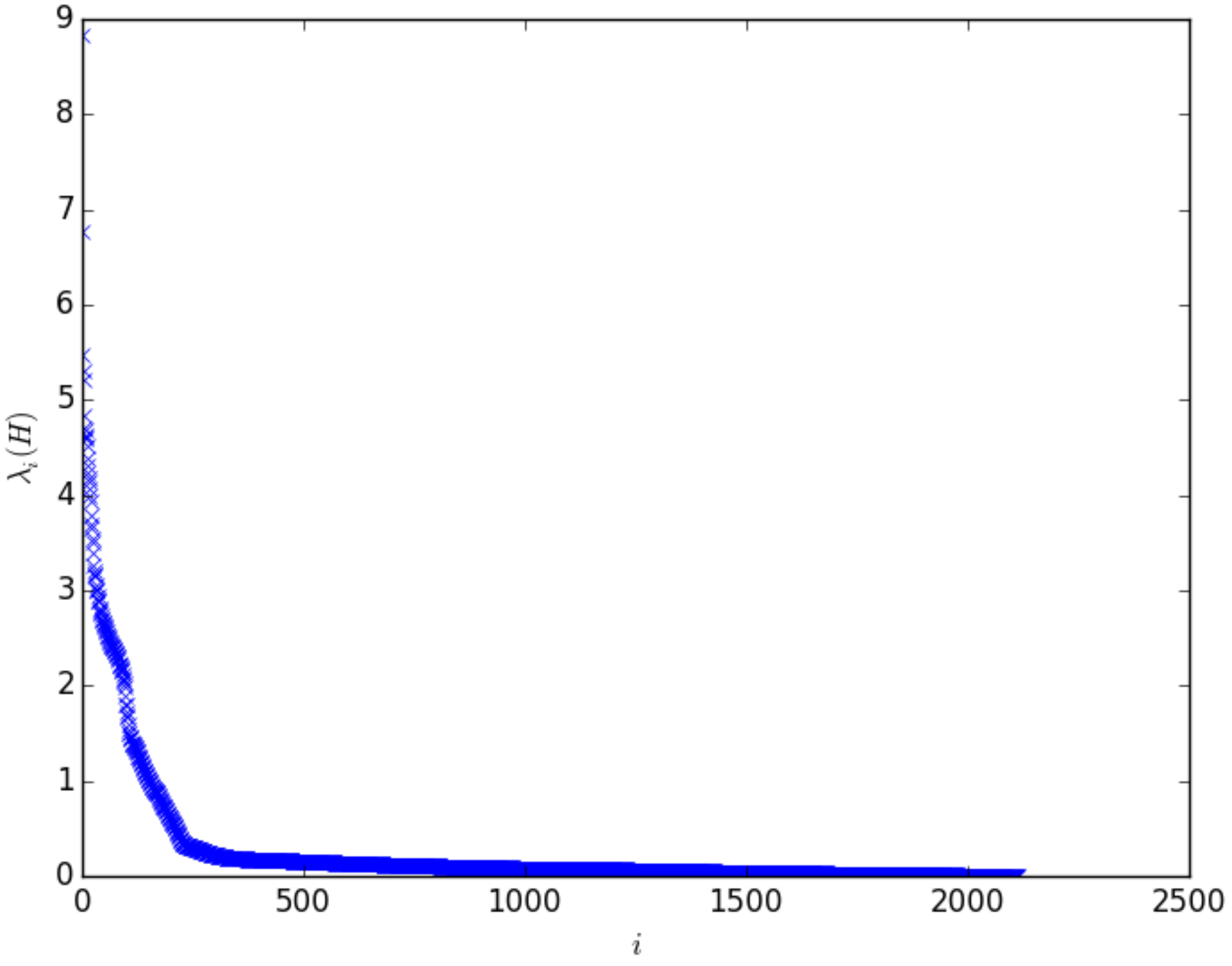

trailing_eigenvector_0
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1.2
-1.4
-1.6
-1.8
-2
-2.2
-2.4
-2.6
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
2.2
2.4
2.6
2.8
3

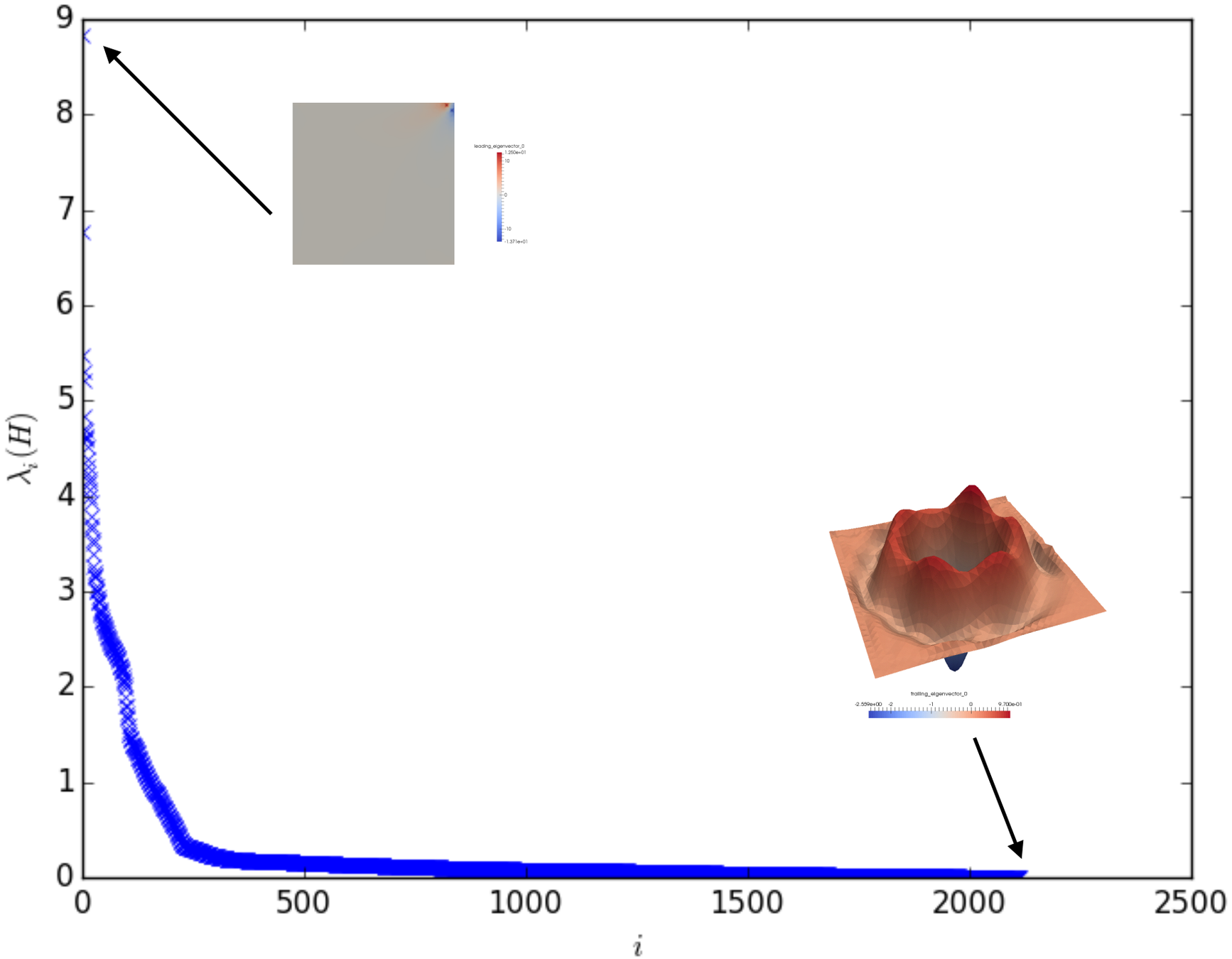
Leading Eigenvectors

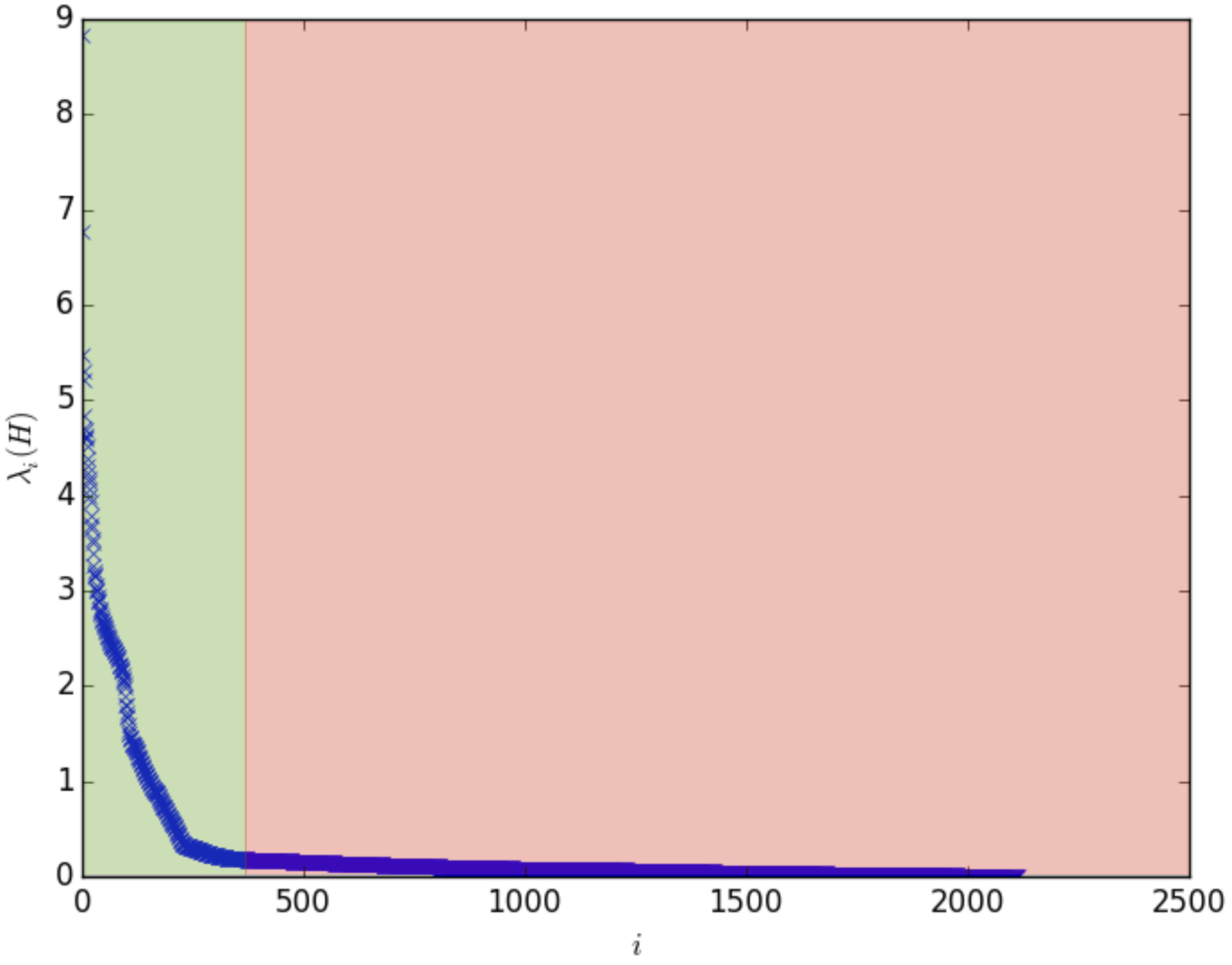
Direction in parameter space *most* constrained by the observations

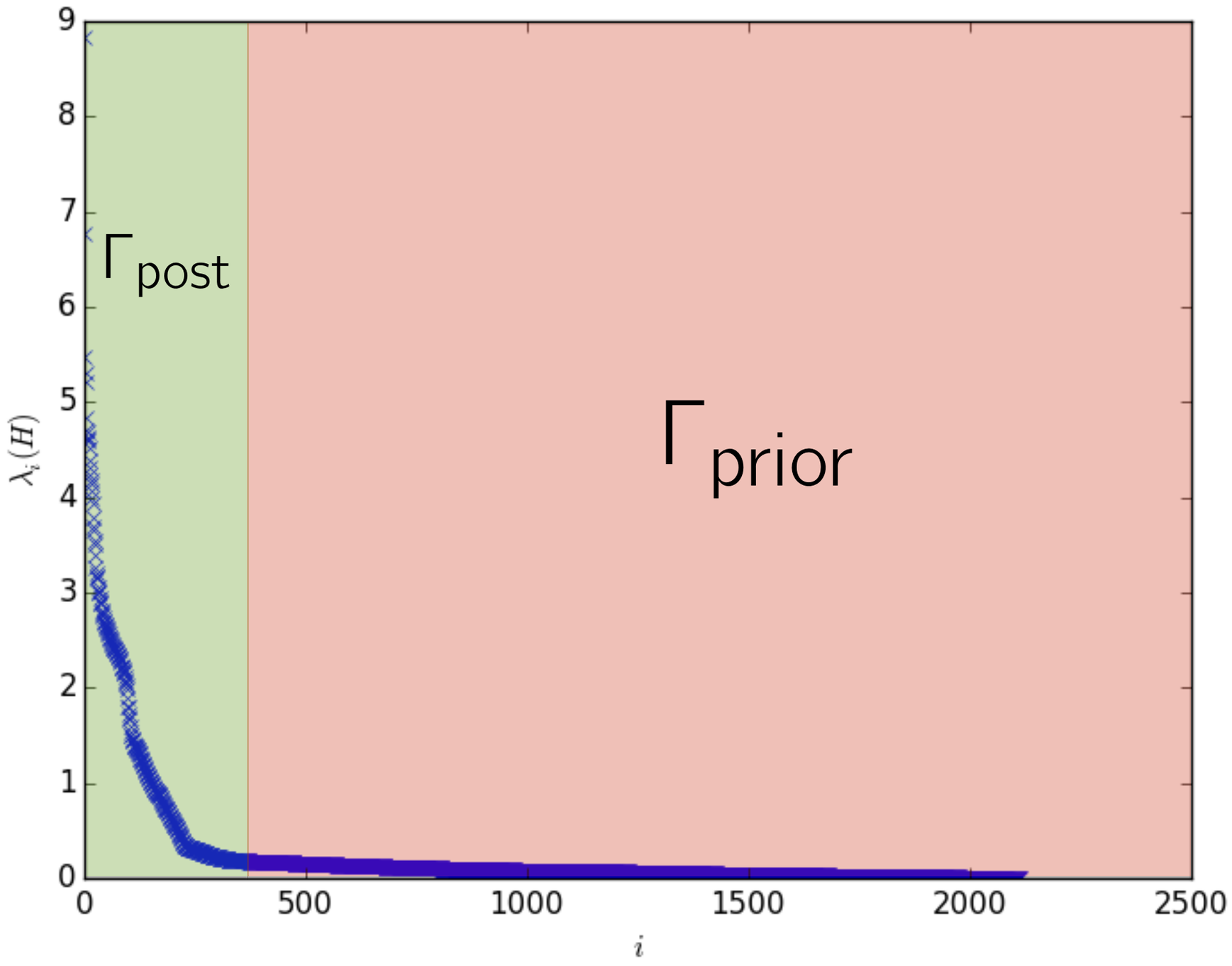

leading_eigenvector_0
1.250e+01
10
0
-10
-1.371e+01


leading_eigenvector_1
1.405e+01
10
0
-10
-1.377e+01


9
8
7
6
5
4
3
2
1
0
0
500
1000
1500
2000
2500
$\lambda_i(H)$
i


9
8
7
6
5
4
3
2
1
0
0
500
1000
1500
2000
2500
$\lambda_i(H)$
i
leading_eigenvector_0
1.250e+01
10
0
-10
-1.371e+01
trailing_eigenvector_0
-2.559e+00
-2
-1
0
9.700e-01


9
8
7
6
5
4
3
2
1
0
$\lambda_i(H)$
0
500
1000
1500
2000
2500
i



Matches trends from Flath et al. p424 for linear parameter to observable maps.

Full Hessian.
4000+ actions.
2000 to calculate H. 2000 to extract.

Partial Hessian.
501 actions for 292 for leading.
209 to calculate H. 292 to extract.

Huge savings in computational cost.
Scales with model dimension.
Implement *low-rank update*.

Summary

- We are developing methods to access uncertainty in the recovered parameters in hyperelastic materials.

- This is done within the framework of Bayesian inversion.

- dolfin-adjoint makes assembling the equations relatively easy, solving them is tougher.

- Next steps: efficient low-rank updates, Hamiltonian MCMC.