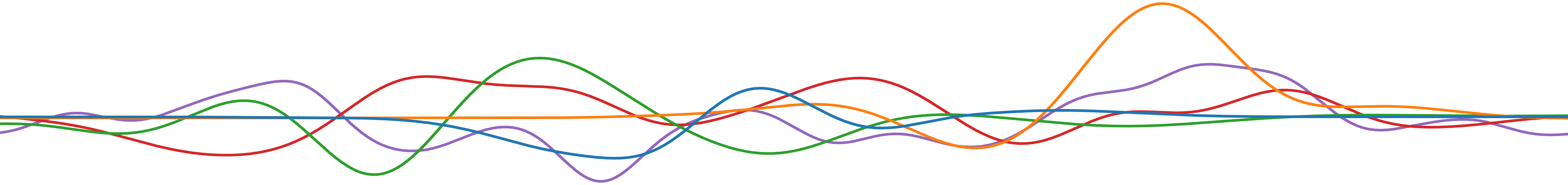
Unfolding with Gaussian Processes

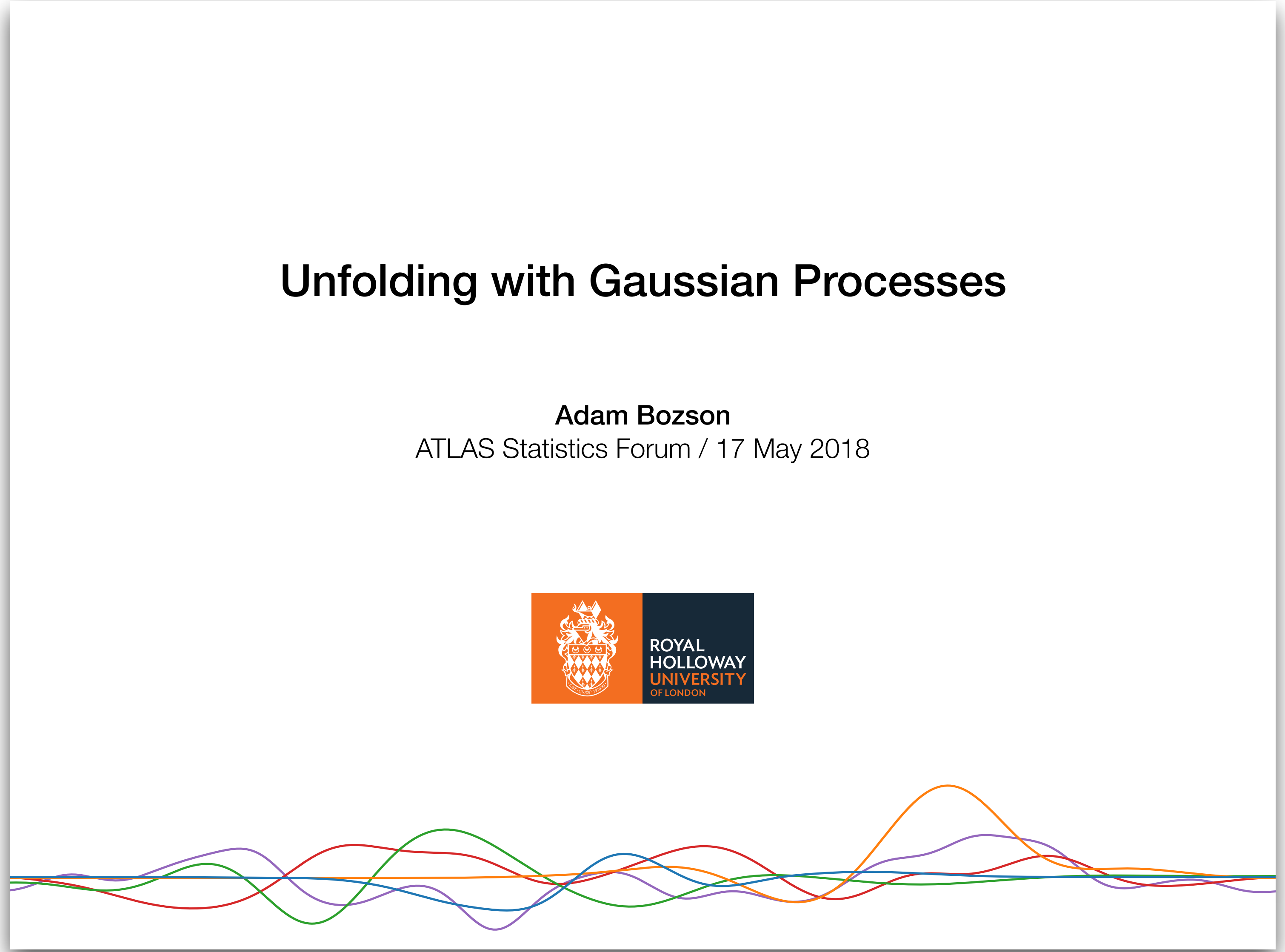
Adam Bozson, Glen Cowan, Francesco Spanò

ATLAS Joint Machine Learning & Statistics Fora Meeting
5 December 2018



Recap

Gave a similar talk in May
indico.cern.ch/event/726207/



Unfolding with Gaussian Processes

Adam Bozson
ATLAS Statistics Forum / 17 May 2018

ROYAL
HOLLOWAY
UNIVERSITY
OF LONDON

Recap

Paper submitted to NIM A
Preprint: arXiv:1811.01242 [physics.data-an]

Unfolding with Gaussian Processes

Adam Bozson*, Glen Cowan, Francesco Spanò

Department of Physics
Royal Holloway, University of London
Egham, Surrey, TW20 0EX, United Kingdom

Abstract

A method to perform unfolding with Gaussian processes (GPs) is presented. Using Bayesian regression, we define an estimator for the underlying truth distribution as the mode of the posterior. We show that in the case where the bin contents are distributed approximately according to a Gaussian, this estimator is equivalent to the mean function of a GP conditioned on the maximum likelihood estimator. Regularisation is introduced via the kernel function of the GP, which has a natural interpretation as the covariance of the underlying distribution. This novel approach allows for the regularisation to be informed by prior knowledge of the underlying distribution, and for it to be varied along the spectrum. In addition, the full statistical covariance matrix for the estimator is obtained as part of the result. The method is applied to two examples: a double-peaked bimodal distribution and a falling spectrum.

Keywords: unfolding, Gaussian process

1. Introduction

Experimental measurements are distorted and biased by detector effects, due to limitations of the measuring instrument and procedures. The need to infer the underlying distribution using the measured data is shared by variety of fields, from astronomy [1] and medical applications [2] to the investigation of the parameters that describe oil well properties [3].

In most of these fields, these techniques are called *deconvolution* or *restoration* [4]. They are used to solve what is defined as the *inverse problem*: to infer an unknown function $f(\boldsymbol{x})$ from the measured data, using knowledge and assumptions of the distortions.

In particle physics such techniques are known as *unfolding* and a variety of methods have been developed for this purpose (for some reviews see Refs. [5, 6, 7]).

In this paper, a novel Bayesian method to perform unfolding in particle physics is proposed. We use an approach on the maximum likelihood (ML) method, and the need for regularisation. In a Bayesian setting, the likelihood is enhanced by prior information so that the ML solution is replaced by the mode of the posterior distribution. Sec. 4 connects the maximum *a posteriori* (MAP) estimator to the solution of a regression problem which conditions prior knowledge encoded in a Gaussian process on the ML solution extracted from data. Example applications are provided in Sec. 5. Finally, we report the conclusions and outlook for future exploration of this method in Sec. 6.

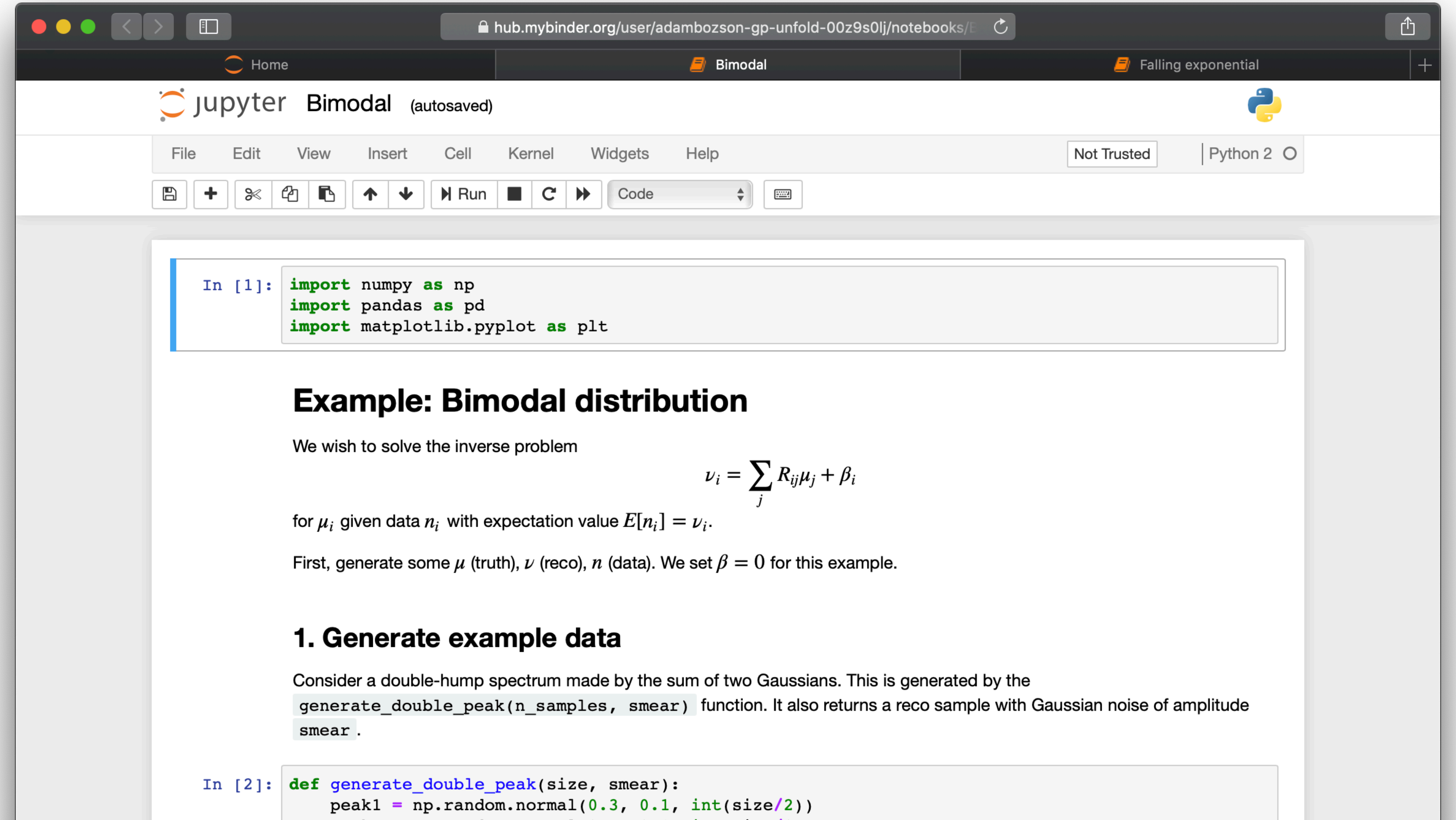
2. Definitions and notation

In particle physics, measured distributions are often reported as populations of bins rather than continuous functions. Therefore the first step we will take is to represent the underlying distributions with discretised bin populations. We note that this process biases the estimated his-

Recap

Code at github.com/adambozson/gp-unfold

launch binder



```
In [1]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
```

Example: Bimodal distribution

We wish to solve the inverse problem

$$\nu_i = \sum_j R_{ij}\mu_j + \beta_i$$

for μ_i given data n_i with expectation value $E[n_i] = \nu_i$.

First, generate some μ (truth), ν (reco), n (data). We set $\beta = 0$ for this example.

1. Generate example data

Consider a double-hump spectrum made by the sum of two Gaussians. This is generated by the `generate_double_peak(n_samples, smear)` function. It also returns a reco sample with Gaussian noise of amplitude `smear`.

```
In [2]: def generate_double_peak(size, smear):
            peak1 = np.random.normal(0.3, 0.1, int(size/2))
```

The unfolding problem

$$f_{\text{reco}}(\boldsymbol{x}) = \int R(\boldsymbol{x}|\boldsymbol{y})\, f_{\text{true}}(\boldsymbol{y})\, \mathrm{d}\boldsymbol{y}$$

$$\nu_i = \int\limits_{\text{bin } i} f_{\text{reco}}(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x} \qquad \Bigg\downarrow \text{Discretise} \qquad \mu_j = \int\limits_{\text{bin } j} f_{\text{truth}}(\boldsymbol{y})\, \mathrm{d}\boldsymbol{y}$$

$$\boldsymbol{\nu} = R\boldsymbol{\mu} + \boldsymbol{\beta}$$

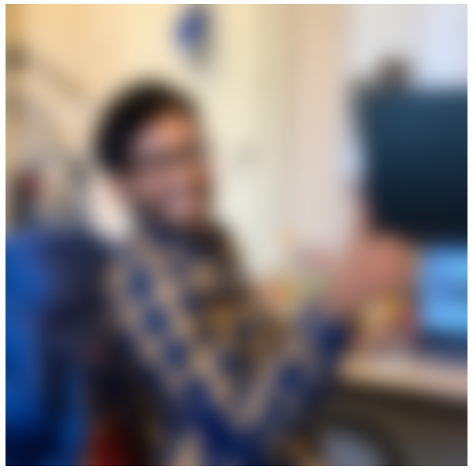
The unfolding problem

$$\boldsymbol{\nu} = R\boldsymbol{\mu} + \cancel{\boldsymbol{\beta}}$$

Background

Set to zero for now

Reco



Truth

Aim: Given data, estimate truth

* Deshan was very happy to feature on this slide

Likelihood

Measure data $\boldsymbol{n}$

with expectation values $E[\boldsymbol{n}] = \boldsymbol{\nu}$

and covariance matrix V

If the data distribution can be approximated as Gaussian

$$\boldsymbol{n} \sim \mathcal{N}\left(\boldsymbol{\nu}, V\right)$$

Then the log-likelihood is

$$\begin{aligned}\log P(\boldsymbol{n}|\boldsymbol{\nu}) &= -\frac{1}{2}\left(\boldsymbol{n} - \boldsymbol{\nu}\right)^{\mathsf{T}} V^{-1}\left(\boldsymbol{n} - \boldsymbol{\nu}\right) + \ldots \\ & \qquad\qquad \downarrow \qquad\qquad\qquad\qquad \searrow \; \boldsymbol{\nu} = R\boldsymbol{\mu} \\ &= -\frac{1}{2}\left(\boldsymbol{n} - R\boldsymbol{\mu}\right)^{\mathsf{T}} V^{-1}\left(\boldsymbol{n} - R\boldsymbol{\mu}\right) + \ldots\end{aligned}$$

Maximum likelihood

The maximum of

$$\log P(\boldsymbol{n}|\boldsymbol{\mu}) = -\frac{1}{2}\left(\boldsymbol{n} - R\boldsymbol{\mu}\right)^{\mathsf{T}} V^{-1}\left(\boldsymbol{n} - R\boldsymbol{\mu}\right)$$

is

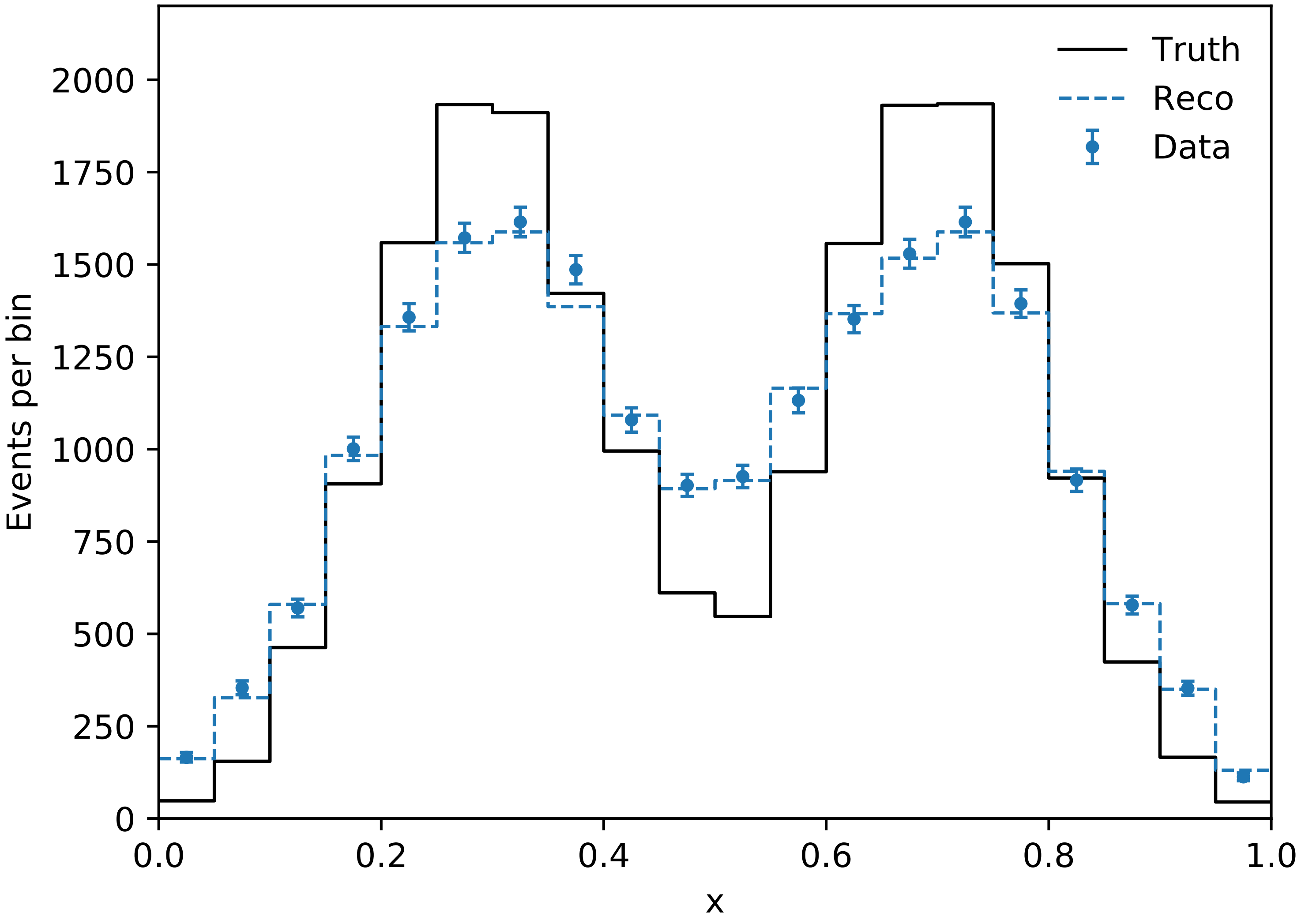
$$\hat{\boldsymbol{\mu}}_{\mathrm{ML}} = R^{-1}\boldsymbol{n}$$

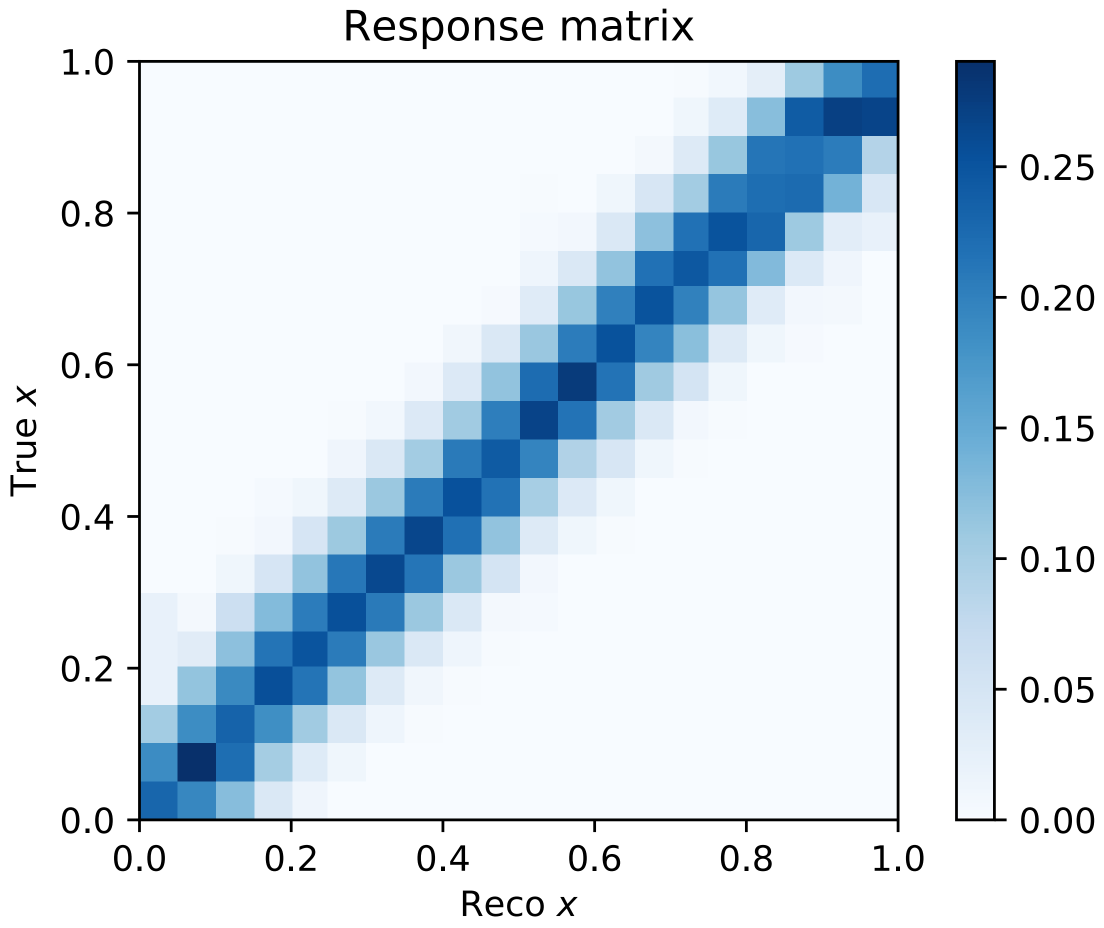
$$U = R^{-1}V\left(R^{-1}\right)^{\mathsf{T}}$$

(Same result for Poisson)

Example

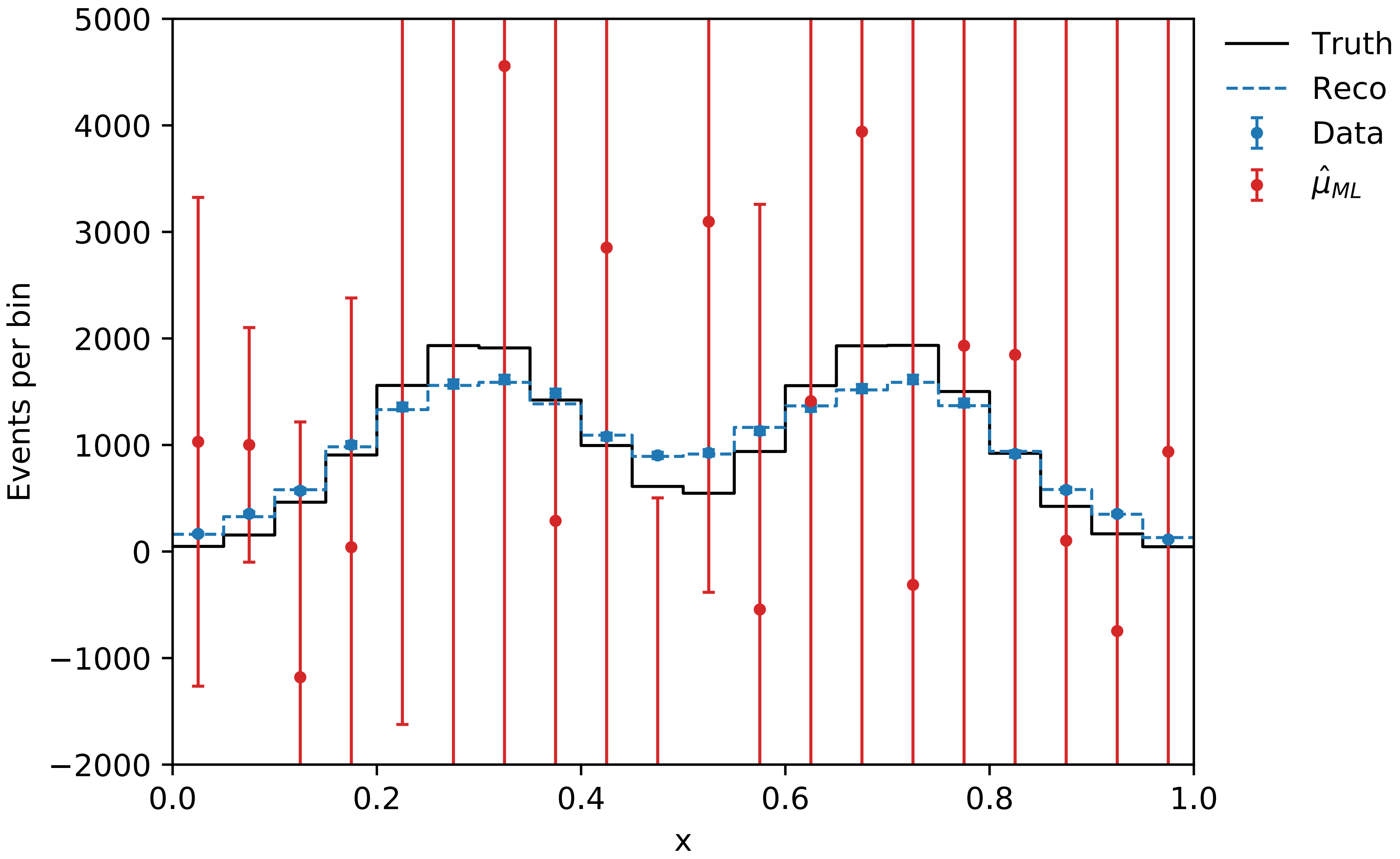
Ingredients



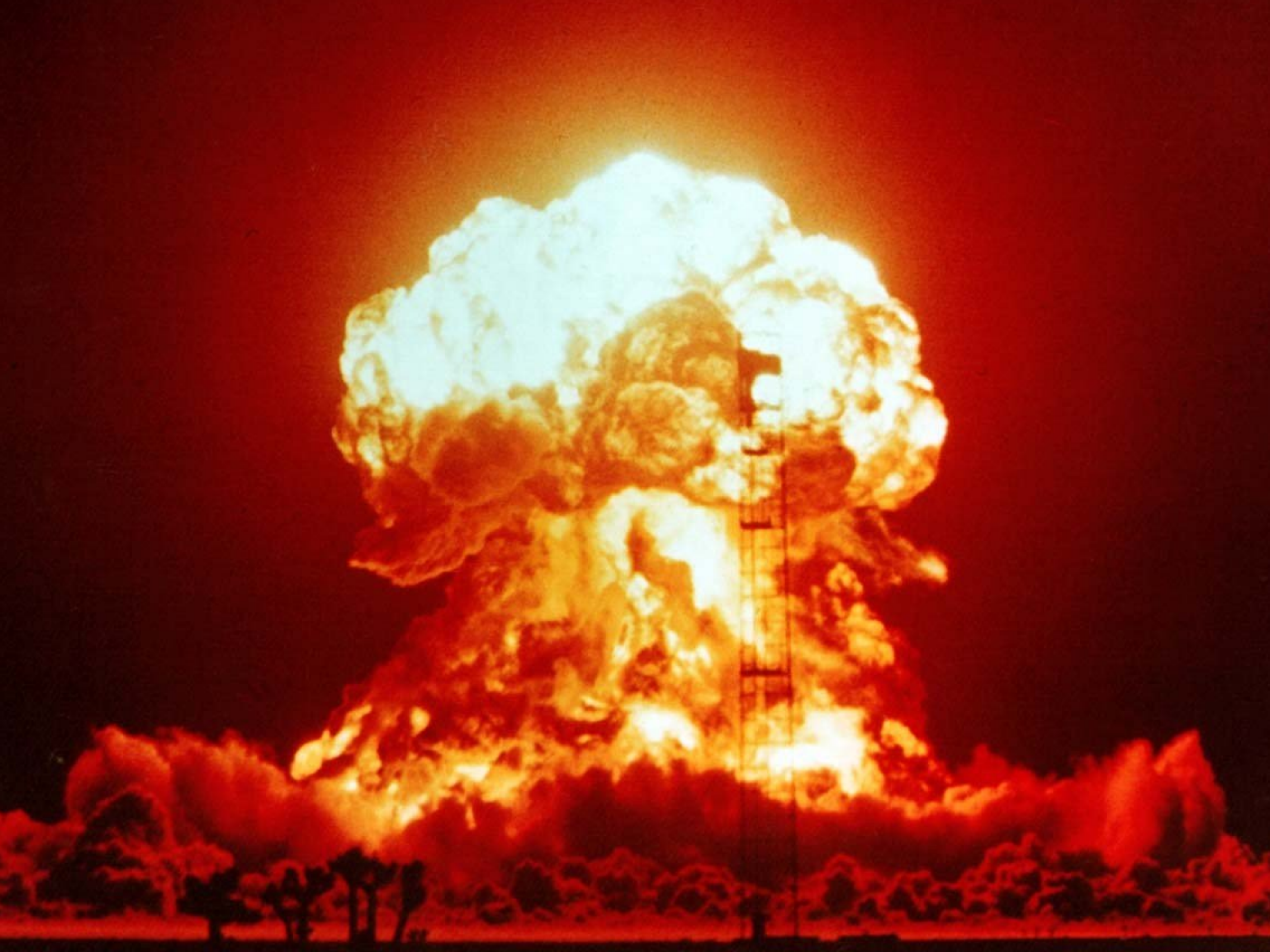


Example

Result

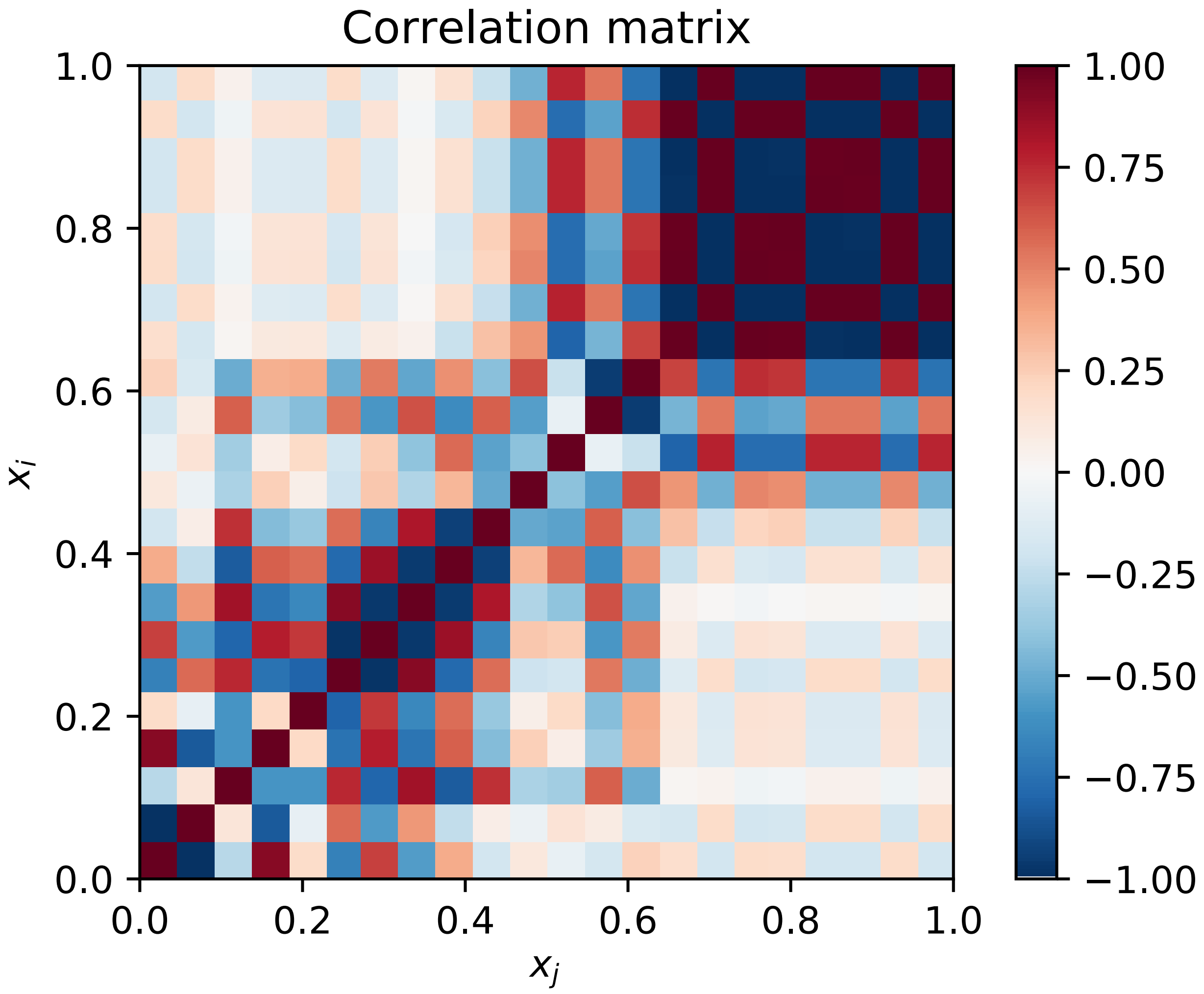


```
mu_ML = scipy.linalg.solve(R, data)
```



The covariance of the ML (both Poisson and Gaussian) estimator is

$$U = R^{-1} V \left(R^{-1}\right)^{\mathsf{T}}$$



The covariance of the ML (both Poisson and Gaussian) estimator is

$$U = R^{-1} V \left(R^{-1} \right)^{\mathsf{T}}$$

Statistical **fluctuations** in the data lead to false **fine structure** (high-frequency oscillations) in the unfolded distribution

The ML estimator is **unbiased**

Regularisation

A common solution is to instead maximise

$$\Phi(\boldsymbol{\mu}) = \alpha \log P(\boldsymbol{n}|\boldsymbol{\mu}) + S(\boldsymbol{\mu})$$

Regularisation parameter
Controls bias vs. variance

Regularisation function
Reduces space of solutions

or an iterative method (Lucy, Richardson), stopping before the ML solution

or a Bayesian method (FBU), conditioning a prior on the data

Common theme: we expect the unfolded distribution to have some smoothness (based on our knowledge of the underlying physics)

Maximum *a posteriori*

Posterior probability given by Bayes' theorem

$$P(\boldsymbol{\mu}|\boldsymbol{n}) = \frac{P(\boldsymbol{n}|\boldsymbol{\mu})\,P(\boldsymbol{\mu})}{P(\boldsymbol{n})}$$

$$\log P(\boldsymbol{\mu}|\boldsymbol{n}) = \log \underset{\swarrow}{P(\boldsymbol{n}|\boldsymbol{\mu})} + \log \underset{\searrow}{P(\boldsymbol{\mu})} + \ldots$$

Likelihood

Prior

All distributions are approximately Gaussian

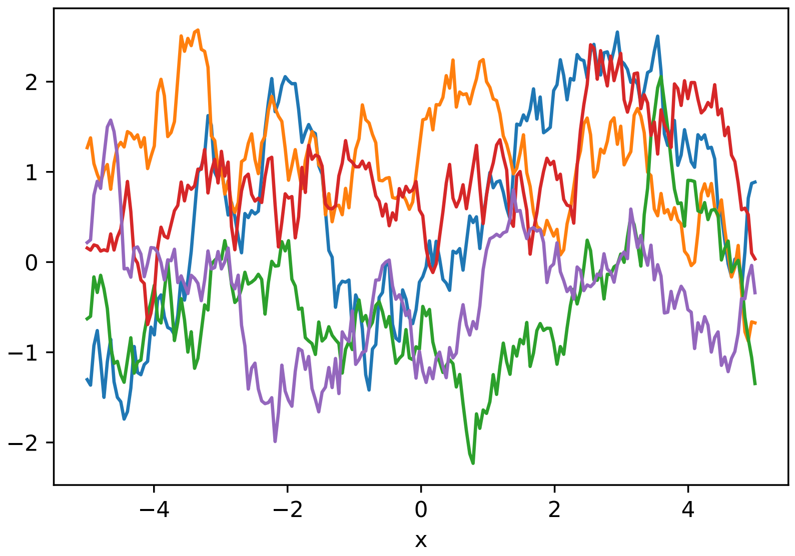
Treat the truth distribution as a **Gaussian process**

$$\boldsymbol{\mu} \sim \mathcal{N}\left(\bar{\boldsymbol{\mu}}, K\right)$$

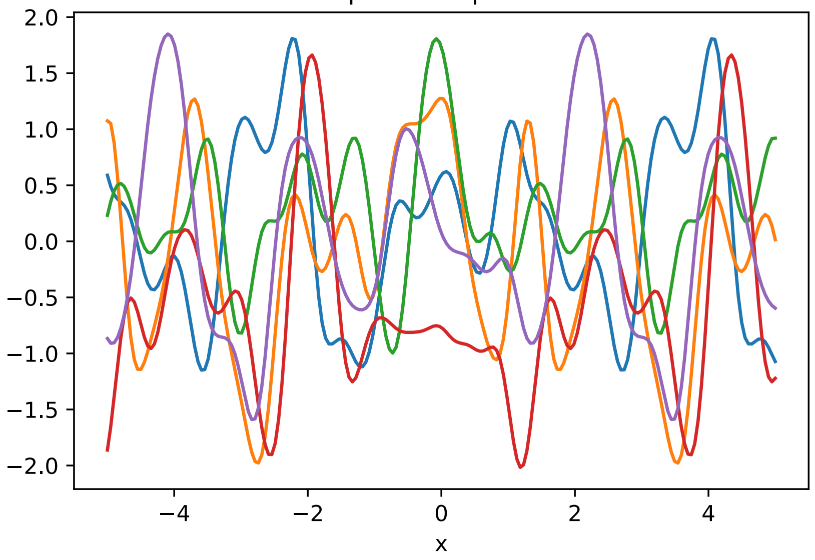
Covariance matrix from **kernel function**

$$K_{ij} = k(y_i, y_j)$$

Priors for Ornstein-Uhlenbeck kernel



Priors for squared exponential kernel



$$\log P(\boldsymbol{\mu}) = -\frac{1}{2}\left(\boldsymbol{\mu} - \bar{\boldsymbol{\mu}}\right)^{\mathsf{T}} K^{-1}\left(\boldsymbol{\mu} - \bar{\boldsymbol{\mu}}\right) + \ldots$$

Maximum *a posteriori*

$$\begin{aligned}\log P(\boldsymbol{\mu}|\boldsymbol{n}) &= \log P(\boldsymbol{n}|\boldsymbol{\mu}) + \log P(\boldsymbol{\mu}) + \ldots \\ &= -\frac{1}{2}\left(\boldsymbol{n} - R\boldsymbol{\mu}\right)^{\mathsf{T}} V^{-1}\left(\boldsymbol{n} - R\boldsymbol{\mu}\right) - \frac{1}{2}\left(\boldsymbol{\mu} - \bar{\boldsymbol{\mu}}\right)^{\mathsf{T}} K^{-1}\left(\boldsymbol{\mu} - \bar{\boldsymbol{\mu}}\right) + \ldots\end{aligned}$$

Use the **mode** of the posterior as an estimator for the unfolded histogram

$$\left.\frac{\mathrm{d}\log P(\boldsymbol{\mu}|\boldsymbol{n})}{\mathrm{d}\boldsymbol{\mu}}\right|_{\boldsymbol{\mu}=\hat{\boldsymbol{\mu}}} = \left(\boldsymbol{n} - R\hat{\boldsymbol{\mu}}\right)^{\mathsf{T}} V^{-1}R + \left(\hat{\boldsymbol{\mu}} - \bar{\boldsymbol{\mu}}\right)^{\mathsf{T}} K^{-1} = \mathbf{0}$$

$$\begin{aligned}\hat{\boldsymbol{\mu}} &= \left[K^{-1} + R^{\mathsf{T}}V^{-1}R\right]^{-1}\left(R^{\mathsf{T}}V^{-1}\boldsymbol{n} + K^{-1}\bar{\boldsymbol{\mu}}\right) \\ &= K\left[K + R^{-1}V(R^{-1})^{\mathsf{T}}\right]^{-1}\left(R^{-1}\boldsymbol{n} - \bar{\boldsymbol{\mu}}\right) + \bar{\boldsymbol{\mu}}\end{aligned}$$

$$\hat{\boldsymbol{\mu}} = K\left[K + \underline{R^{-1}V(R^{-1})^{\mathsf{T}}}\right]^{-1}\left(\underline{R^{-1}\boldsymbol{n}} - \bar{\boldsymbol{\mu}}\right) + \bar{\boldsymbol{\mu}}$$

$$U = R^{-1}V\left(R^{-1}\right)^{\mathsf{T}} \qquad \hat{\boldsymbol{\mu}}_{\mathrm{ML}} = R^{-1}\boldsymbol{n}$$

Remember the **ML** estimator?

The **MAP estimator** is the **mean** of a **Gaussian process regressor** using the **ML estimator** as training points

Since the posterior distribution is Gaussian (product of Gaussians), the mode is equal to the mean

GP regression

Start from a **prior** over functions

$$f(\boldsymbol{x}) \sim \mathcal{GP}\big(m(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}')\big)$$

Given observations $\boldsymbol{y}$ at X, we want to estimate $\boldsymbol{f}_*$ at X_*

Condition the prior on the observations via Bayes' rule

$$\boldsymbol{f}_* | \, \boldsymbol{y} \sim \mathcal{N}\left(\bar{\boldsymbol{f}}_*, \Sigma_*\right), \quad \text{where}$$

$$\bar{\boldsymbol{f}}_* = K_*^{\mathsf{T}} \left[K + \Sigma\right]^{-1} (\boldsymbol{y} - \boldsymbol{m}) + \boldsymbol{m}_*,$$

$$\Sigma_* = K_{**} - K_*^{\mathsf{T}} \left[K + \Sigma\right]^{-1} K_*$$

See e.g. gaussianprocess.org/gpml

GP unfolding

Posterior mean for GP regression

$$\boldsymbol{f}_* | \boldsymbol{y} \sim \mathcal{N}\left(\bar{\boldsymbol{f}}_*, \Sigma_*\right), \quad \text{where}$$

$$\bar{\boldsymbol{f}}_* = K_*^{\mathsf{T}} \left[K + \Sigma\right]^{-1} (\boldsymbol{y} - \boldsymbol{m}) + \boldsymbol{m}_*,$$

$$\Sigma_* = K_{**} - K_*^{\mathsf{T}} \left[K + \Sigma\right]^{-1} K_*$$

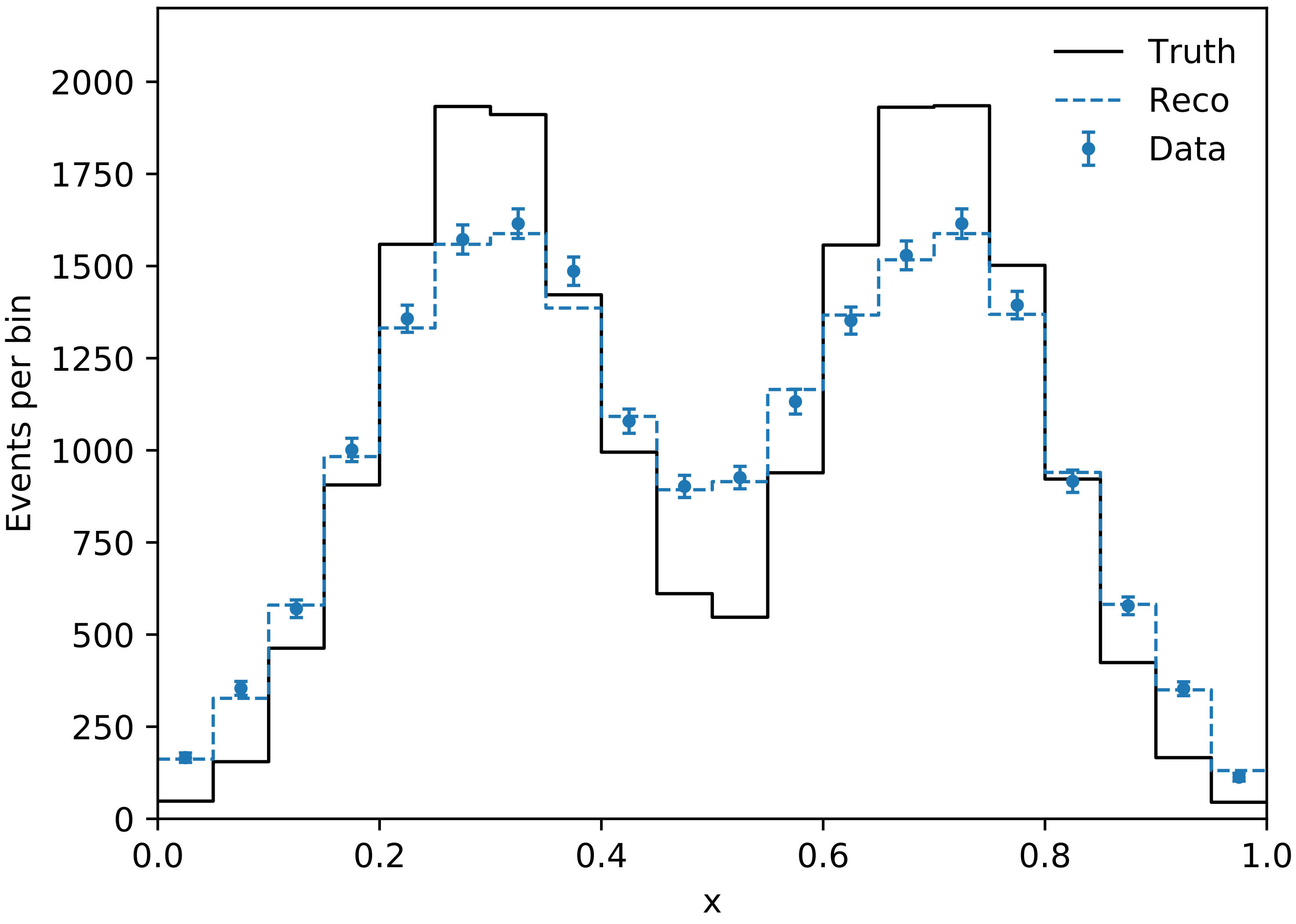
So we can generalise the MAP result

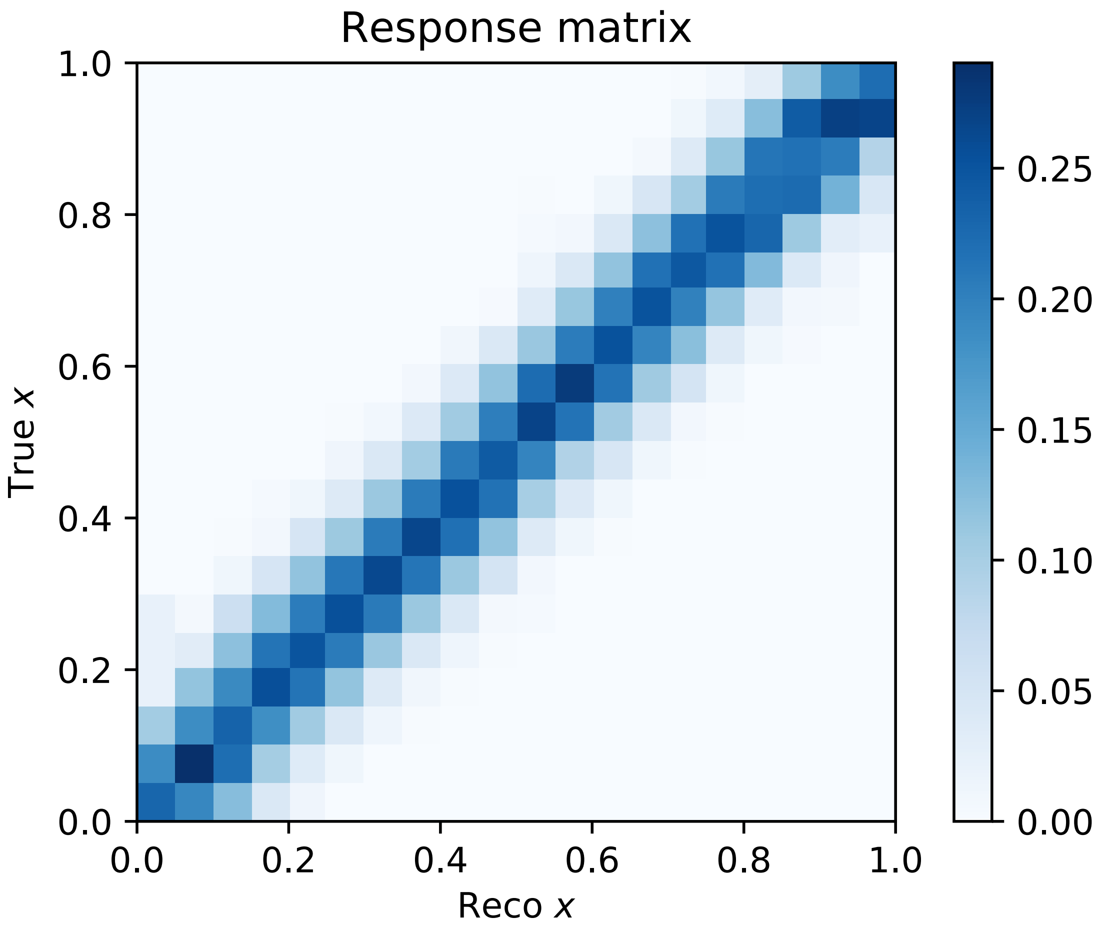
$$\hat{\boldsymbol{\mu}}_* = K_*^{\mathsf{T}} \left[K + U\right]^{-1} \left(R^{-1}\boldsymbol{n} - \bar{\boldsymbol{\mu}}\right) + \bar{\boldsymbol{\mu}}_*$$

$$\Sigma_* = K_{**} - K_*^{\mathsf{T}} \left[K + U\right]^{-1} K_*$$

Example

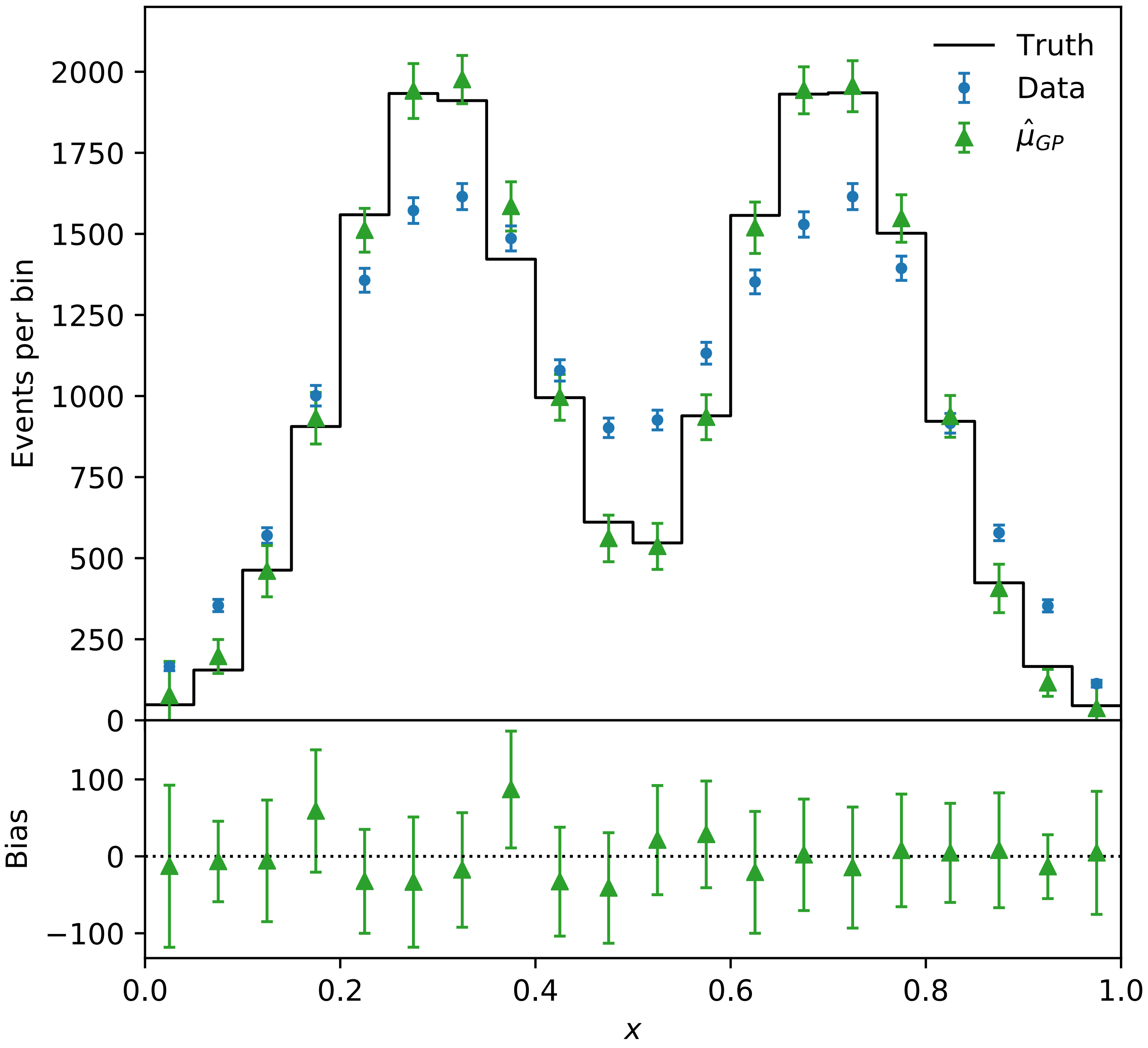
Ingredients





Example

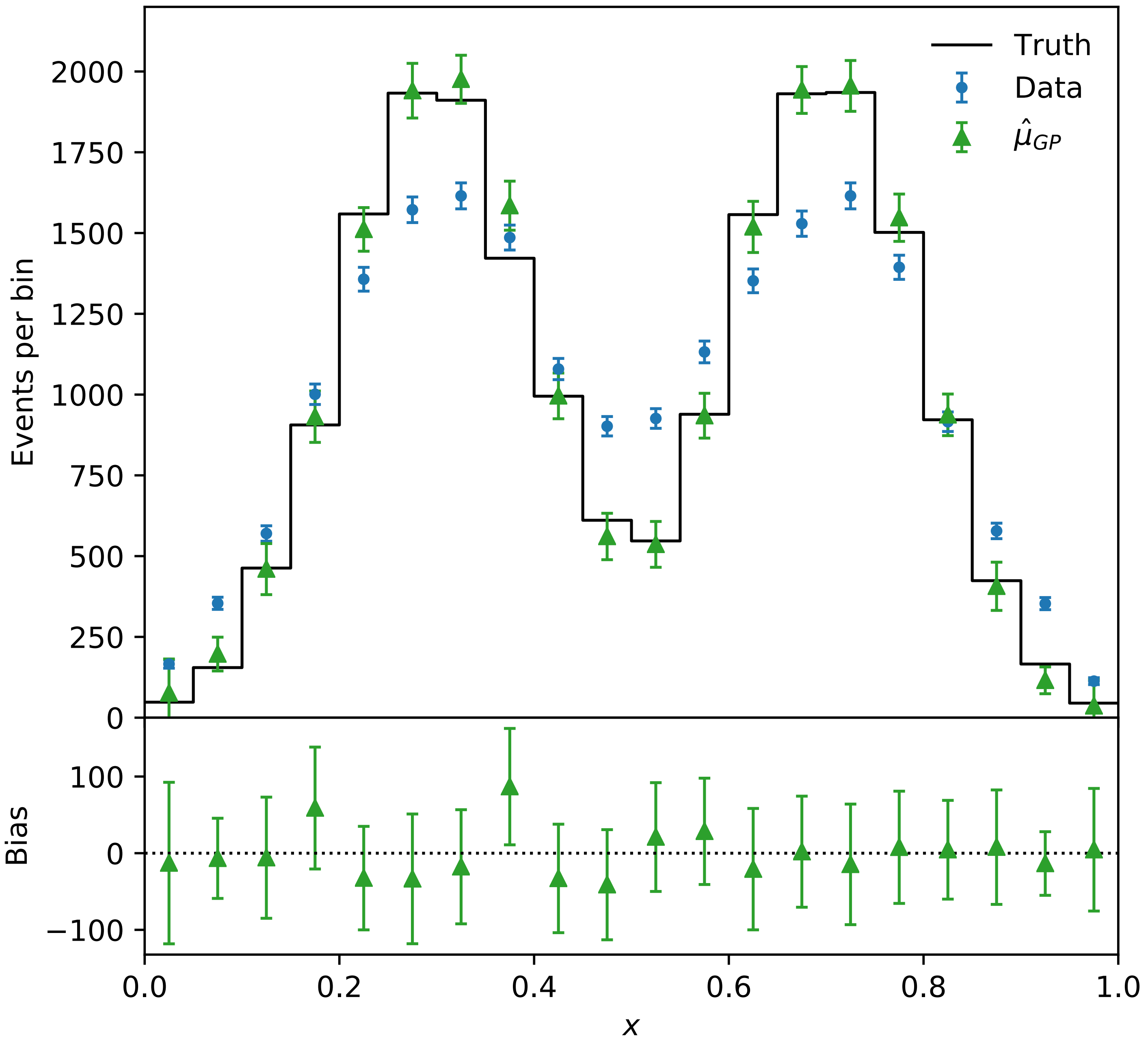
Result

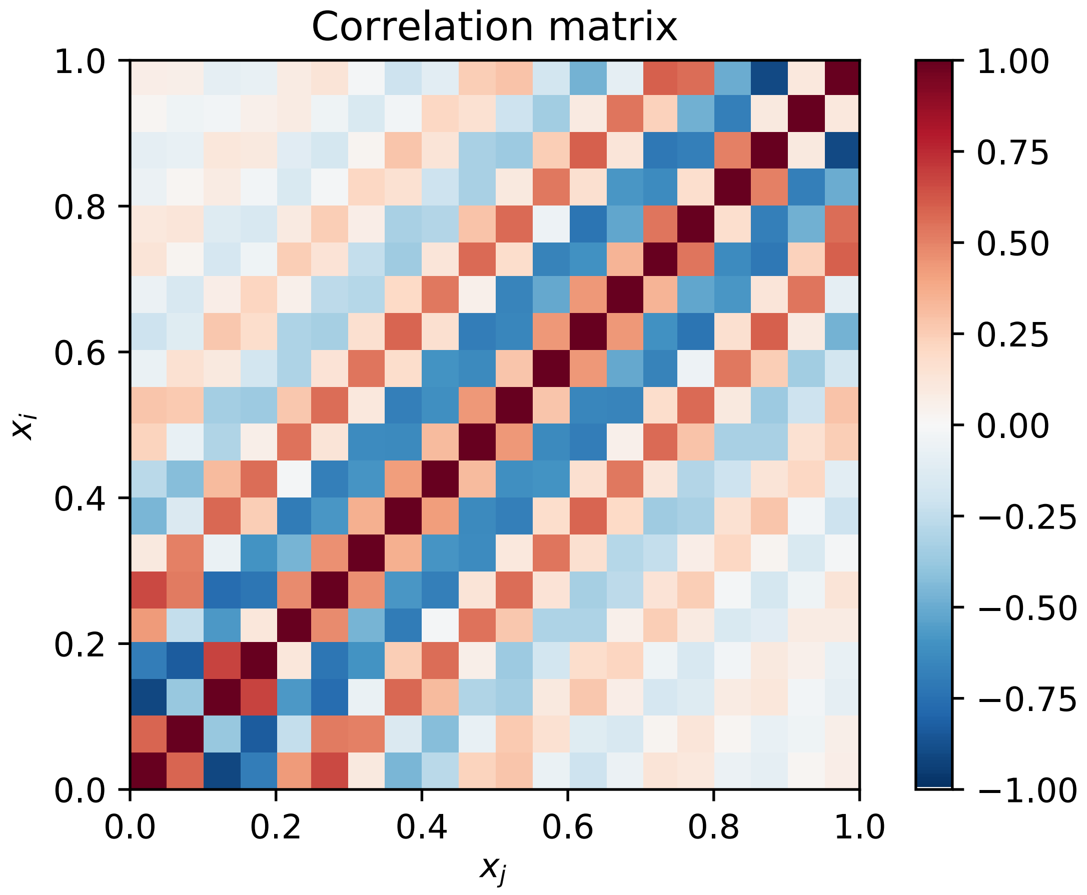


```
U = np.dot(scipy.linalg.inv(R),
           np.dot(np.diag(data),
                  scipy.linalg.inv(R).T))

K = kernel(X, X, params)
alpha = solve(np.dot(R, K+U), data)

mu = np.dot(K, alpha)
cov = K - np.dot(K, solve(K+U, Ks.T))
```

Example

Result





Kernel

The kernel controls the smoothness of the solution

$$k(x, x') = A \exp\left(-\frac{(x - x')^2}{2l^2}\right)$$

Bias/variance

Length scale

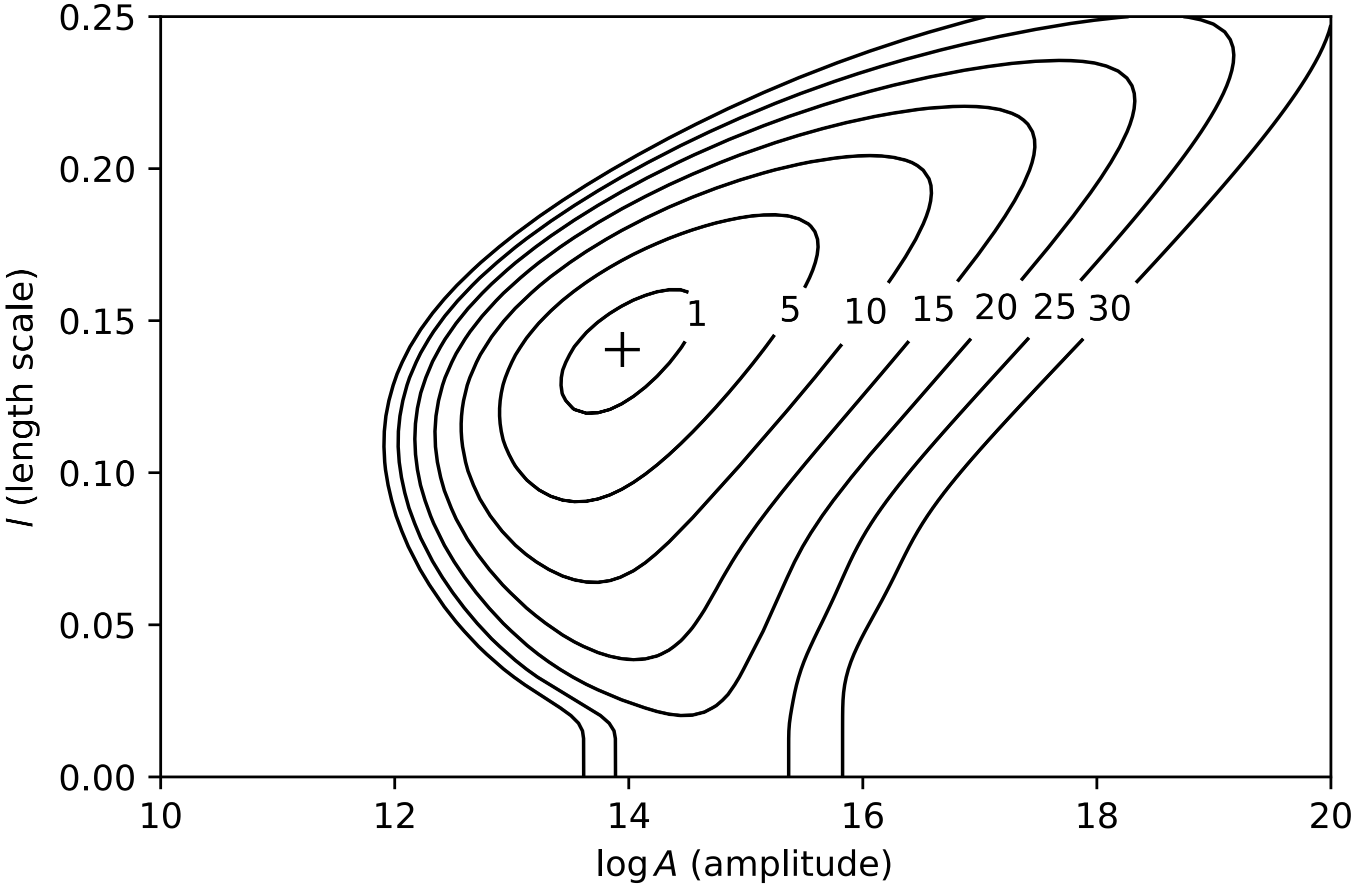
Can use physically-motivated kernels
(e.g. JES/PDF uncertainties: Gibbs kernel)

How to optimise the hyperparameters?

$$\log P(\boldsymbol{n}|\boldsymbol{\theta}) = -\frac{1}{2}\left(R^{-1}\boldsymbol{n} - \bar{\boldsymbol{\mu}}\right)^{\mathsf{T}}\left[K_{\boldsymbol{\theta}} + U\right]^{-1}\left(R^{-1}\boldsymbol{n} - \bar{\boldsymbol{\mu}}\right) - \frac{1}{2}\log|K_{\boldsymbol{\theta}} + U| + \ldots$$

Regularisation can be varied along the spectrum with a non-stationary kernel

Kernel



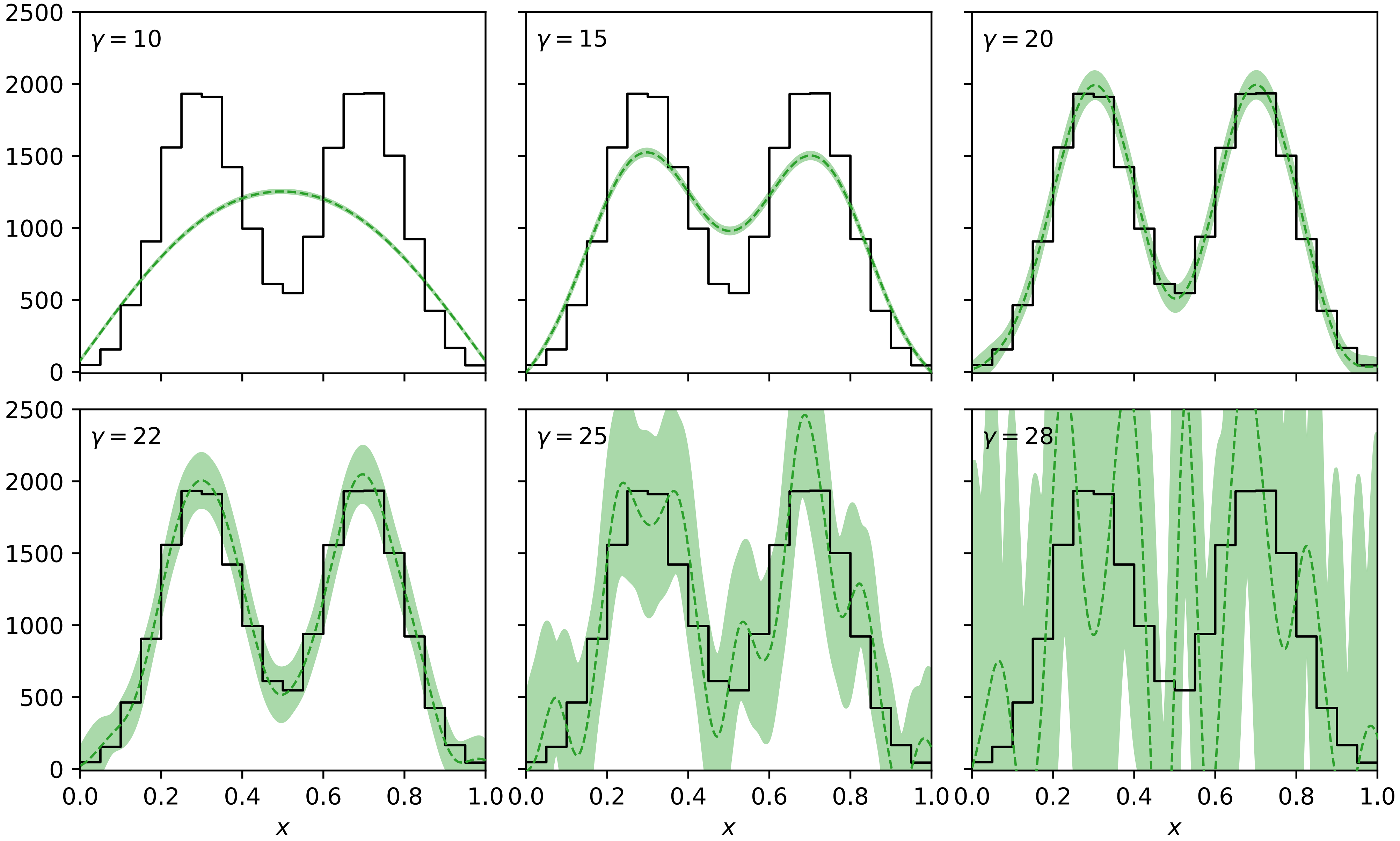
How to optimise the hyperparameters?

$$\log P(\boldsymbol{n}|\boldsymbol{\theta}) = -\frac{1}{2}\left(R^{-1}\boldsymbol{n} - \bar{\boldsymbol{\mu}}\right)^{\mathsf{T}}\left[K_{\boldsymbol{\theta}} + U\right]^{-1}\left(R^{-1}\boldsymbol{n} - \bar{\boldsymbol{\mu}}\right) - \frac{1}{2}\log|K_{\boldsymbol{\theta}} + U| + \ldots$$

Regularisation can be varied along the spectrum with a non-stationary kernel

Varying regularisation

$$k(r) = \frac{e^{\gamma}}{12}(2r^3 - 3Rr^2 + R^3) \text{ where } r = |x - x'|$$



Conclusion

Unfolding can be performed with a GP regressor

The MAP estimator is the mean of a GP using the ML estimator as training points

The kernel controls the regularisation

Preprint: arXiv:1811.01242 [physics.data-an]

Code: github.com/adambozson/gp-unfold