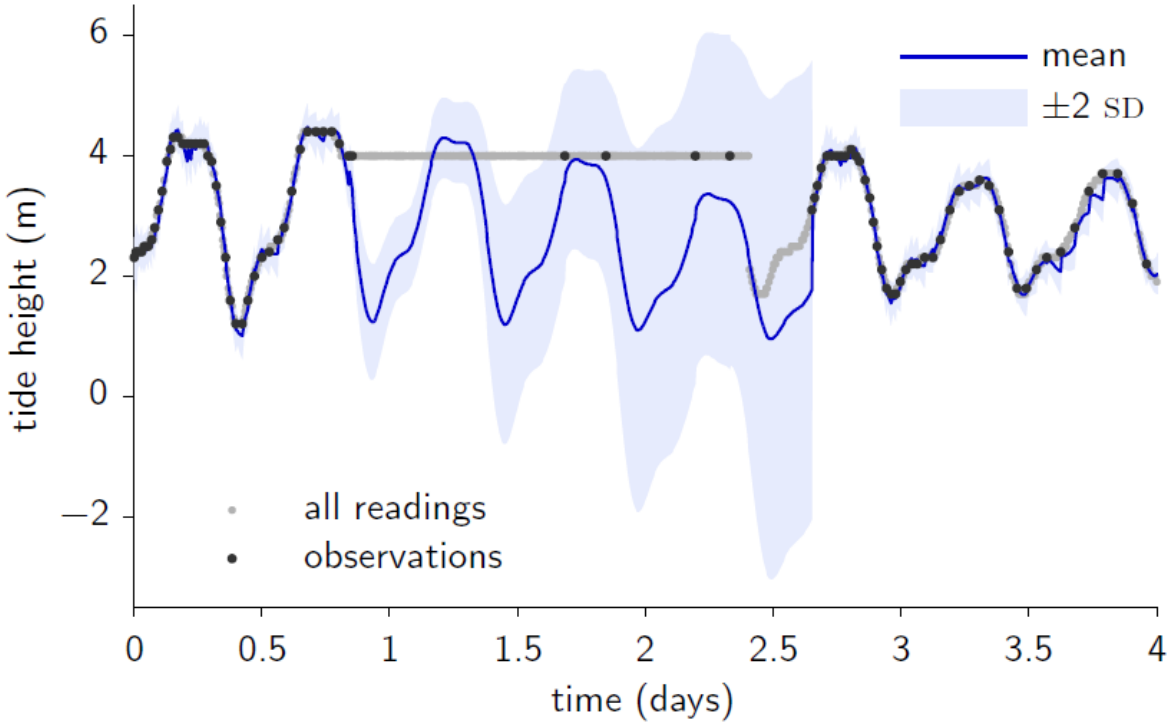
Gaussian Processes for Sequential Prediction

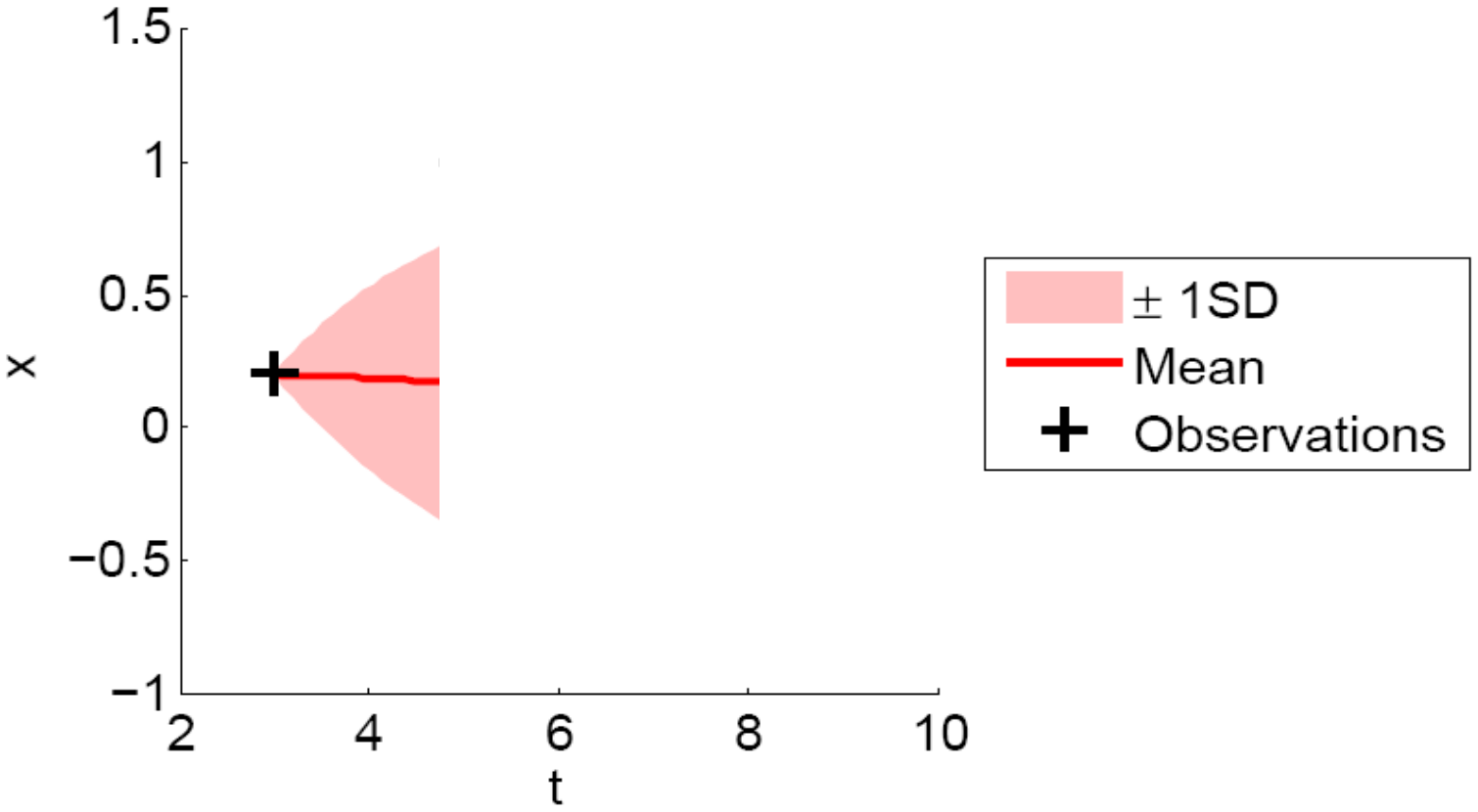
Michael A. Osborne
Machine Learning Research Group
Department of Engineering Science
University of Oxford

UNIVERSITY OF OXFORD

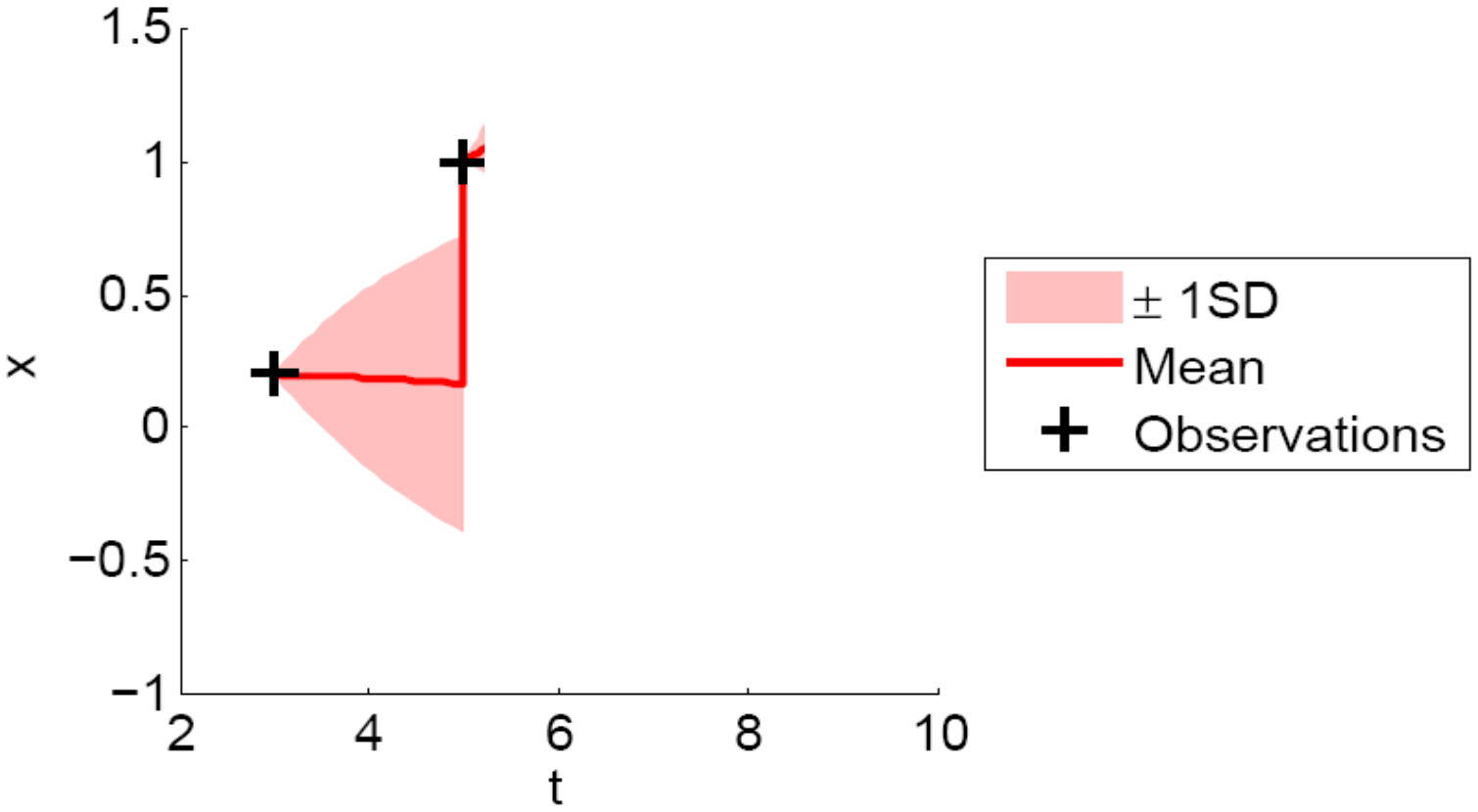
Gaussian processes are useful for sequential data, such as time-series and tracking applications. In particular, Gaussian processes can be used for active data selection (choosing the most informative data) in such systems.



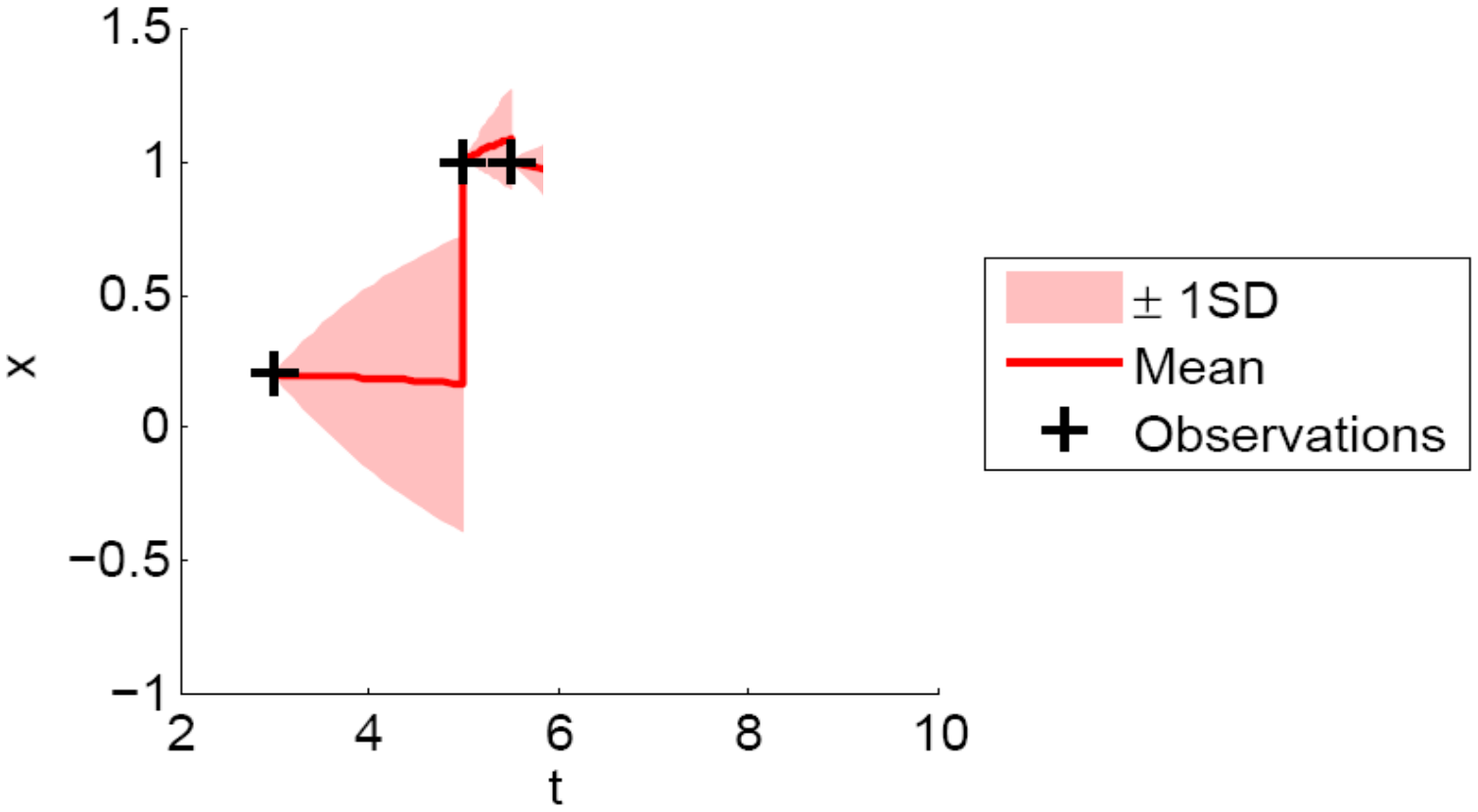
We often want to address functions of time, using Gaussian processes for tracking.



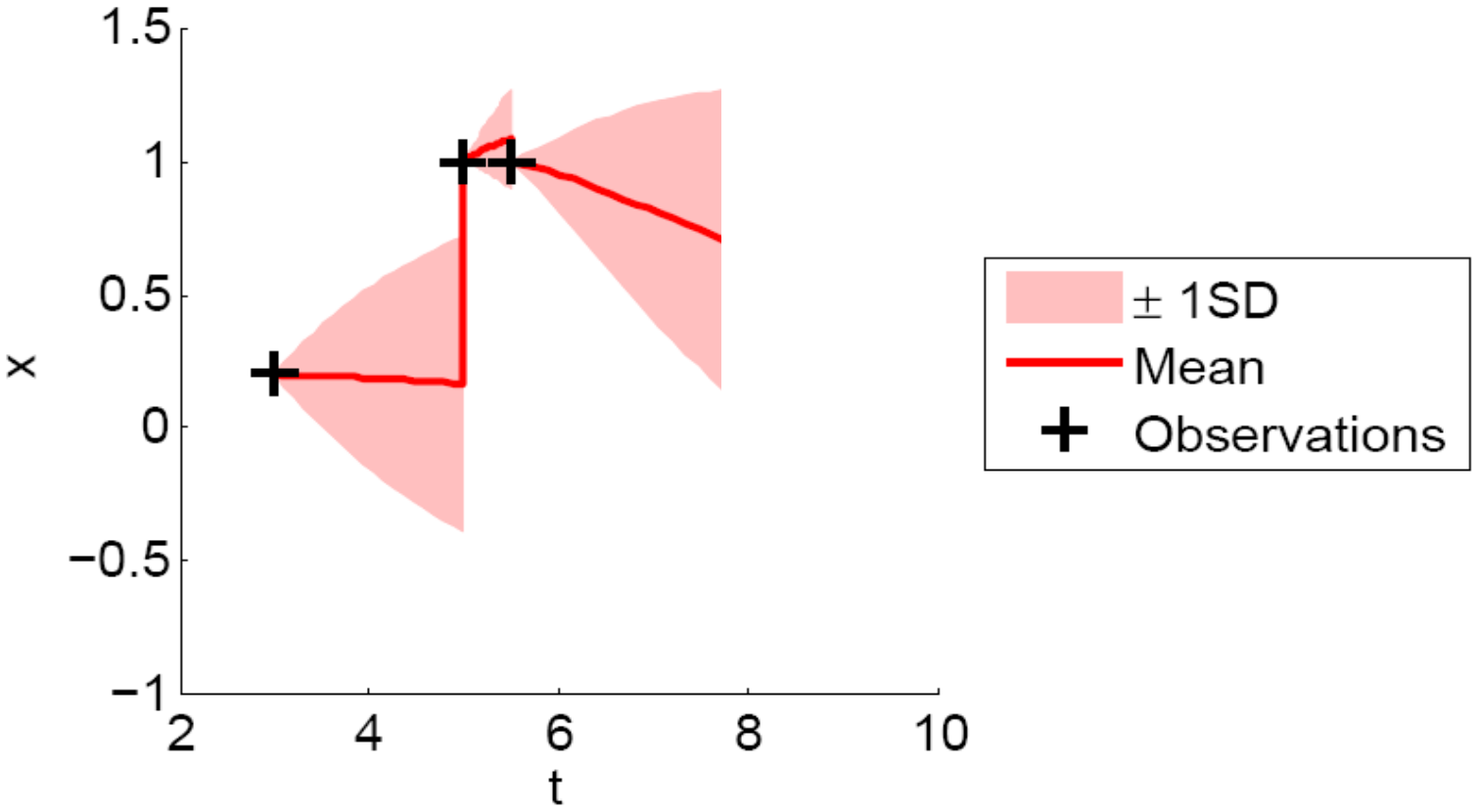
We often want to address functions of time, using Gaussian processes for tracking.



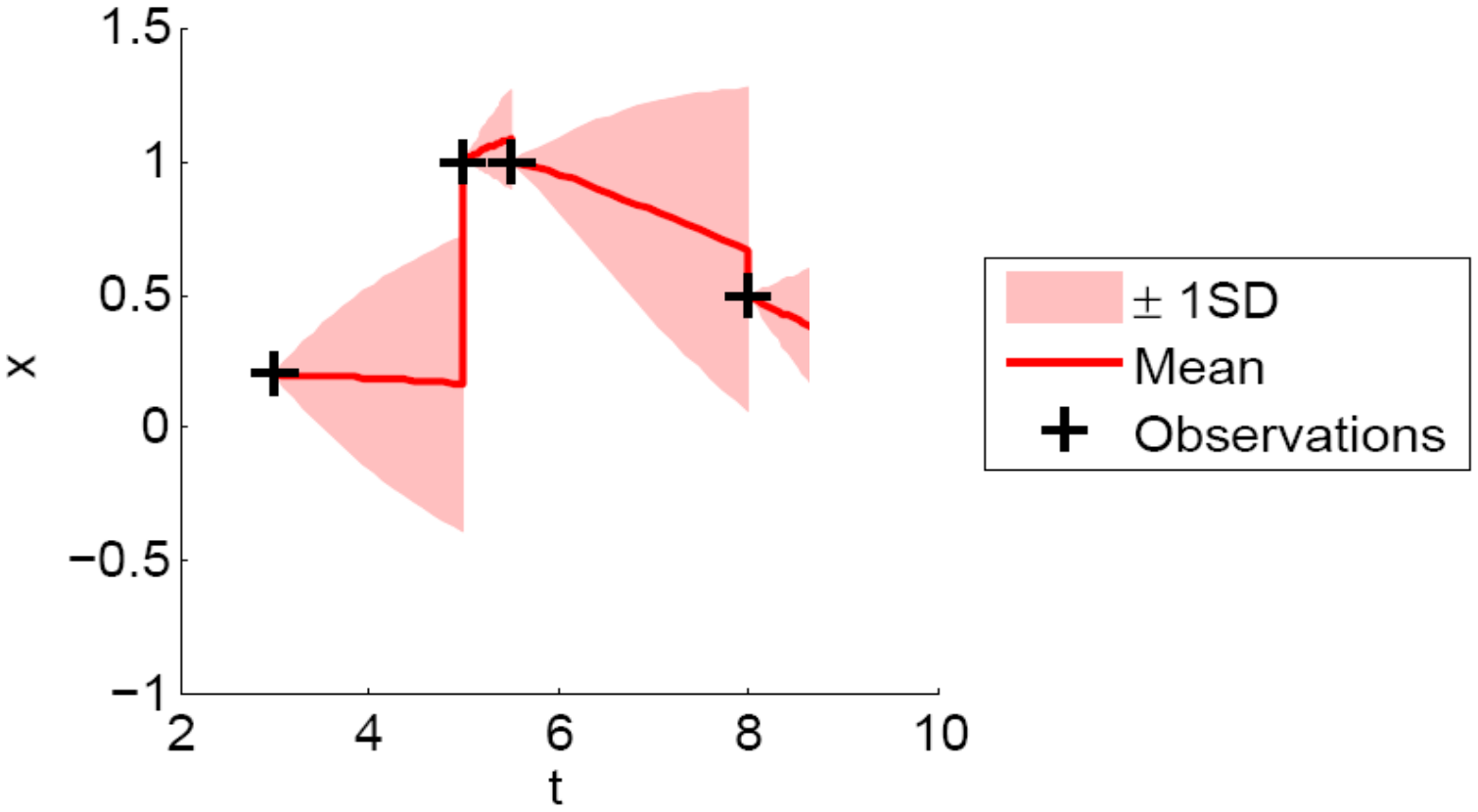
We often want to address functions of time, using Gaussian processes for tracking.



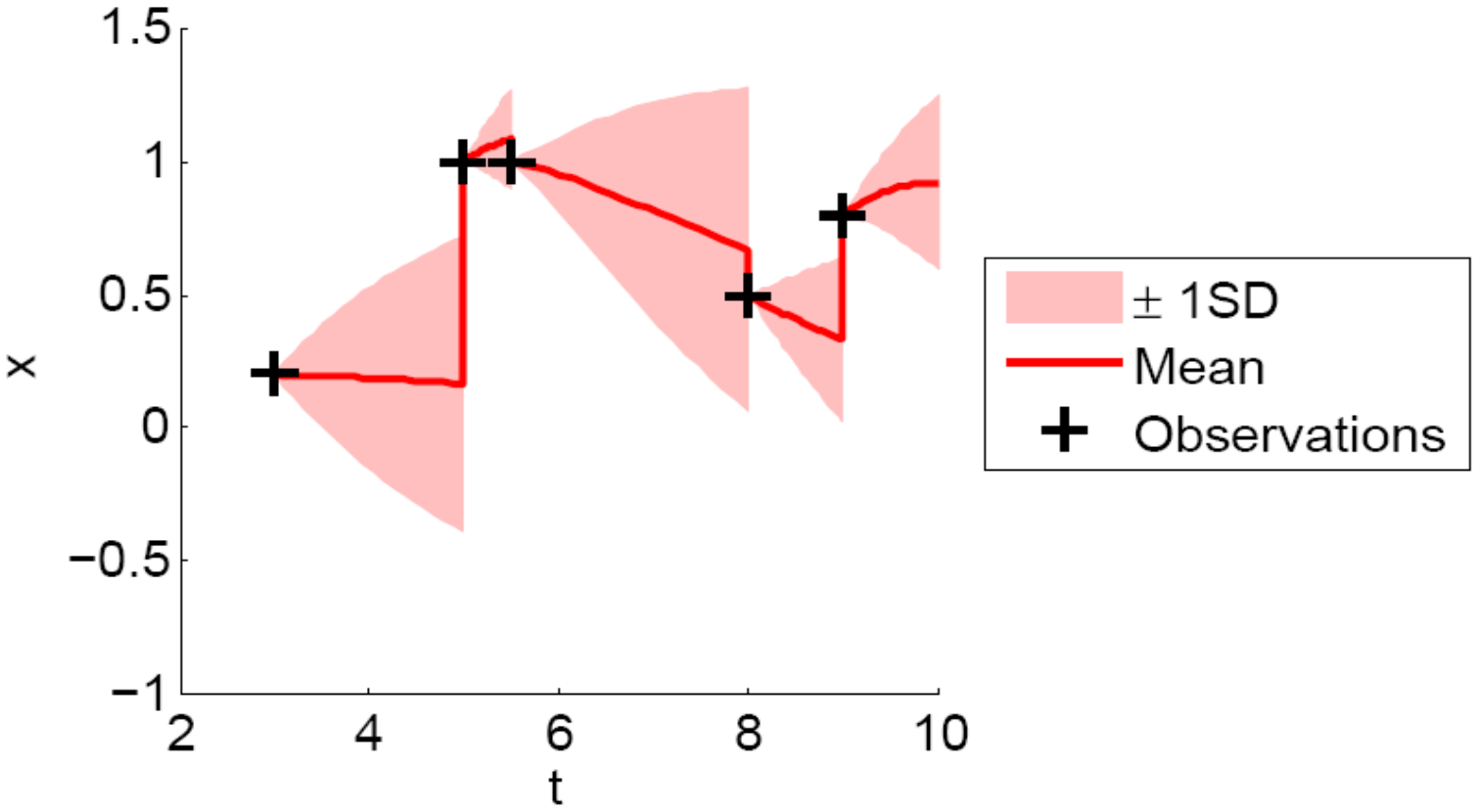
We often want to address functions of time, using Gaussian processes for tracking.



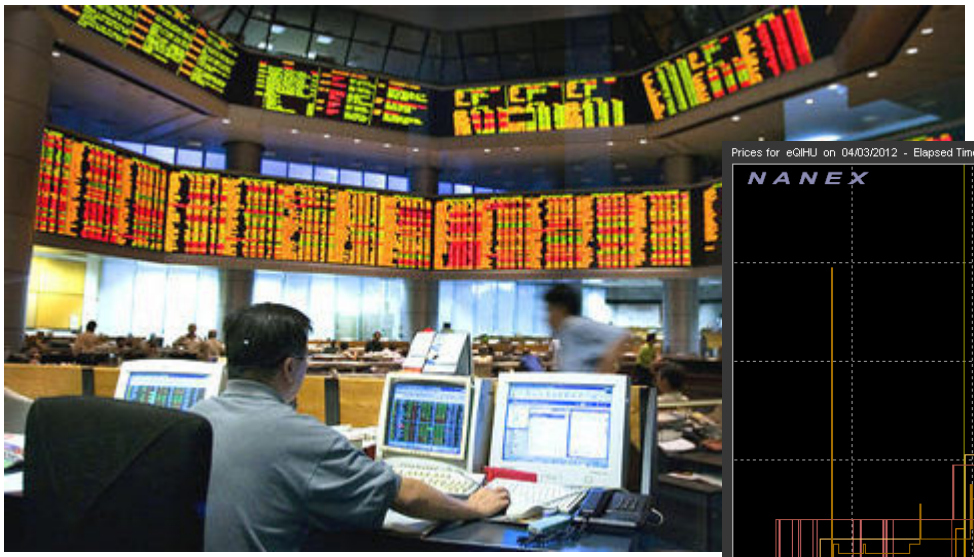
We often want to address functions of time, using Gaussian processes for tracking.

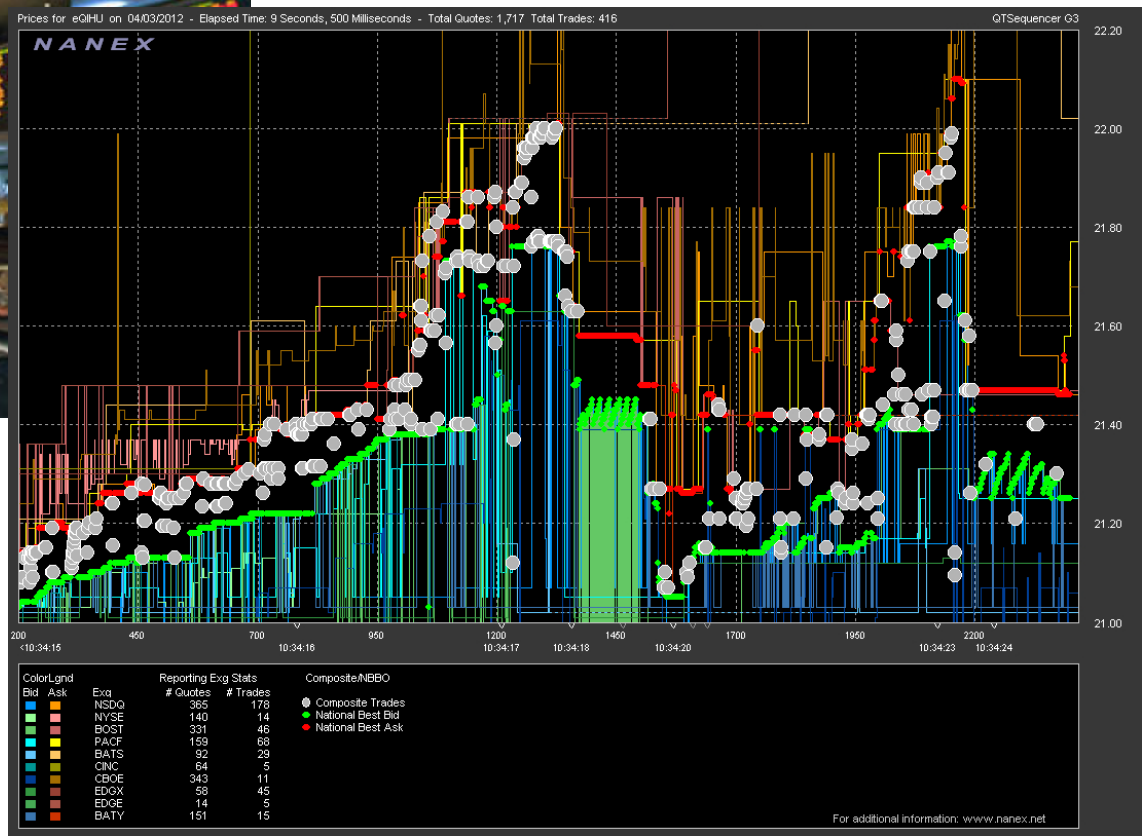


We often want to address functions of time, using Gaussian processes for tracking.



Time-series demand computational efficiency: data arrives rapidly, and must be responded to promptly.



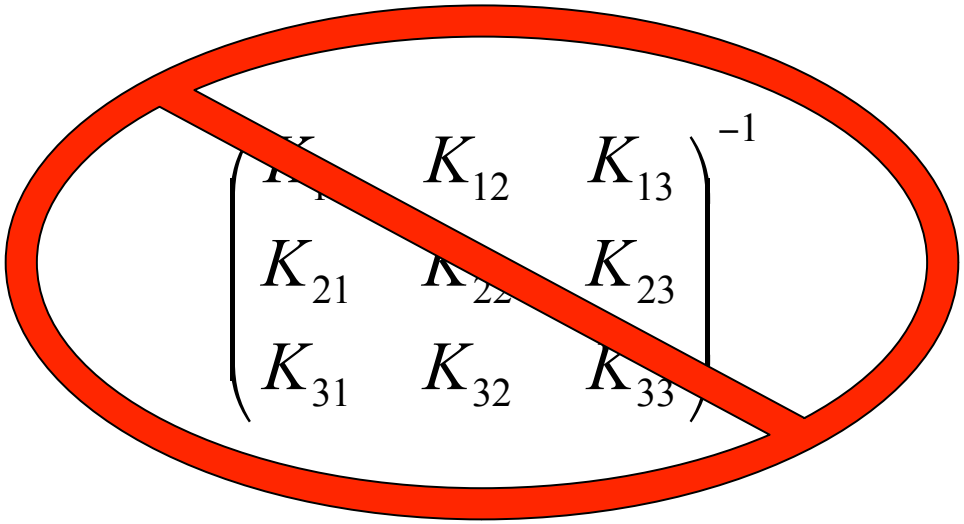


UNIVERSITY OF OXFORD

Gaussian process inference requires evaluating

$$p(\mathbf{f}_*|\mathbf{f}) = \mathcal{N}\left(\mathbf{f}_*|\mathbf{K}_{*,\mathbf{f}}\mathbf{K}_{\mathbf{f},\mathbf{f}}^{-1}\mathbf{f}, \mathbf{K}_{*,*} - \mathbf{K}_{*,\mathbf{f}}\mathbf{K}_{\mathbf{f},\mathbf{f}}^{-1}\mathbf{K}_{\mathbf{f},*}\right)$$

but you should never actually invert a matrix.



Inversion is slow, $O(n^3)$ in matrix size n.
Inversion is also unstable; conditioning errors are significant.

The Cholesky factorisation of a positive semi-definite matrix K is relatively fast ($1/3\ \mathrm{O}(n^3)$ in matrix size n) and more numerically stable.

$$K = R^{\mathrm{T}} R$$

$$R = \mathrm{chol}(K) = \begin{pmatrix} R_{11} & R_{12} & \cdots & R_{1n} \\ 0 & R_{22} & \cdots & R_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & R_{nn} \end{pmatrix}$$

The upper triangular Cholesky factor can then be stored and used to solve $v = K\,x$ for x very quickly ($O(n^2)$ in matrix size n) by back substitution.

$$v = Kx$$

$$v = R^{\mathrm{T}} x'$$

$$x' = Rx$$

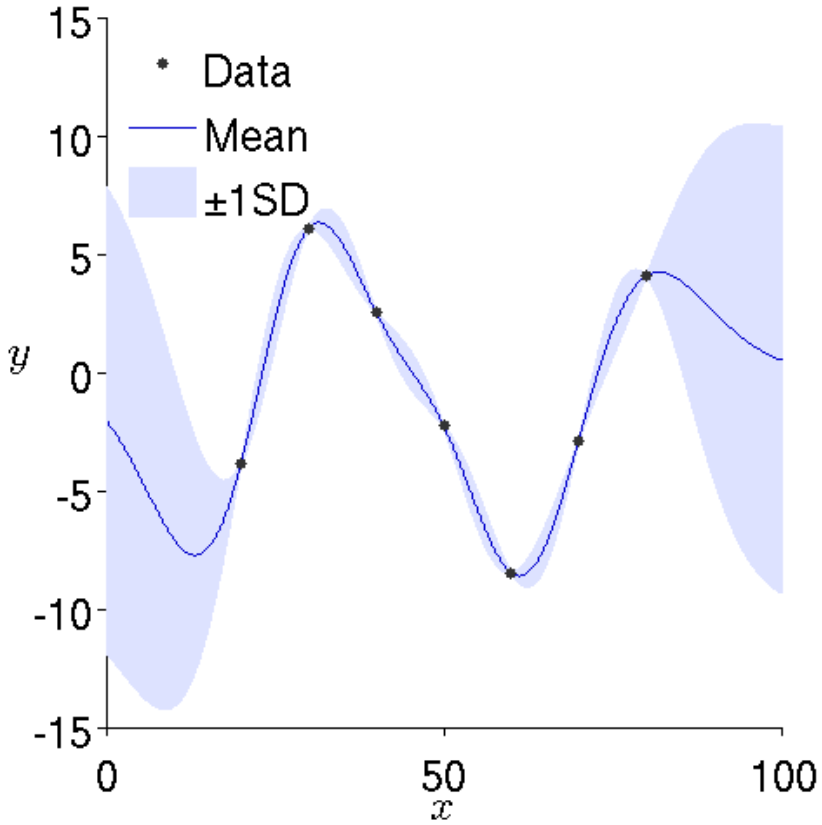
$$\begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix} = \begin{pmatrix} R_{11} & R_{12} & \cdots & R_{1n} \\ 0 & R_{22} & \cdots & R_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & R_{nn} \end{pmatrix} \begin{pmatrix} x'_1 \\ x'_2 \\ \vdots \\ x'_n \end{pmatrix}$$

A symmetric matrix K is Toeplitz if it can be written as

$$K = \begin{pmatrix} k_1 & k_2 & k_3 & k_4 & \cdots & k_n \\ k_2 & k_1 & k_2 & k_3 & & \\ k_3 & k_2 & k_1 & k_2 & & \\ k_4 & k_3 & k_2 & k_1 & & \\ \vdots & & & & \ddots & \\ k_n & & & & & k_1 \end{pmatrix}$$

If K is Toeplitz, there exists a very efficient method to solve $v = K\,x$ for x ($\mathrm{O}(4n^2)$ in matrix size n).

A Gaussian process has a Toeplitz covariance matrix if we have linearly spaced observations and a stationary covariance function: this is common for time-series.

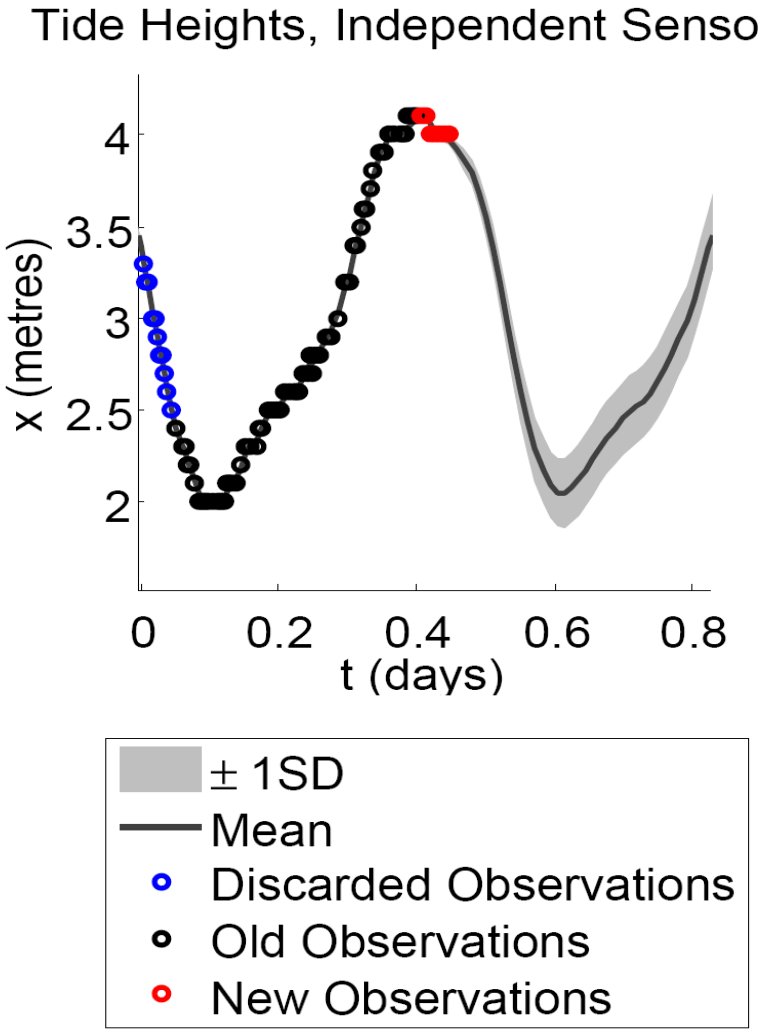


$$K = \begin{pmatrix} 100 & 60.7 & 13.5 & 1.11 & 0.03 & 0 & 0 \\ 60.7 & 100 & 60.7 & 13.5 & 1.11 & 0.03 & 0 \\ 13.5 & 60.7 & 100 & 60.7 & 13.5 & 1.11 & 0.03 \\ 1.11 & 13.5 & 60.7 & 100 & 60.7 & 13.5 & 1.11 \\ 0.03 & 1.11 & 13.5 & 60.7 & 100 & 60.7 & 13.5 \\ 0 & 0.03 & 1.11 & 13.5 & 60.7 & 100 & 60.7 \\ 0 & 0 & 0.03 & 1.11 & 13.5 & 60.7 & 100 \end{pmatrix}$$

If a very large covariance matrix doesn't have Toeplitz structure, we may wish to attempt sparsification, which also simplifies solving.

$$K = \begin{pmatrix} K_{11} & K_{12} & 0 & 0 & \cdots & 0 \\ K_{21} & K_{22} & K_{23} & 0 & & \\ 0 & K_{32} & K_{33} & K_{34} & & \\ 0 & 0 & K_{43} & K_{44} & & \\ \vdots & & & & \ddots & \\ 0 & & & & & K_{nn} \end{pmatrix}$$

There are many ways to sparsify our data; the simplest involve selecting a subset. Windowing represents a reasonable way to do this for sequential data.



If we already have the Cholesky factor

$$R_{11} = \mathrm{chol}(K_{11}),$$

we can efficiently determine the updated factor

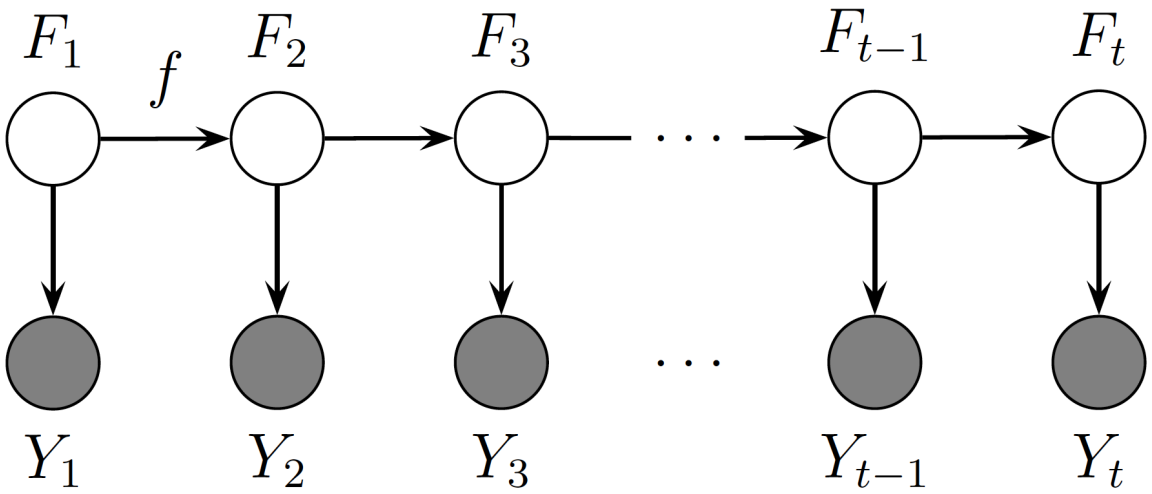
$$\begin{pmatrix} R_{11} & R_{12} \\ 0 & R_{22} \end{pmatrix} = \mathrm{chol}\left(\begin{pmatrix} K_{11} & K_{12} \\ K_{12} & K_{22} \end{pmatrix} \right),$$

in $\mathrm{O}(n^2)$. Similar is true for other types of Cholesky updates and downdates, and for solutions based upon them. A Toeplitz update is probably also possible.

The Kalman filter is a Gaussian process with a special covariance function, one that gives a sparse precision matrix. This allows efficient $O(n)$ computation.

$$K = (\text{ugly})$$

$$K^{-1} = \begin{pmatrix} 2 & -1 & 0 & 0 & \cdots \\ -1 & 2 & -1 & 0 & \\ 0 & -1 & 2 & -1 & \\ 0 & 0 & -1 & 2 & \\ \vdots & & & & \ddots \end{pmatrix}$$

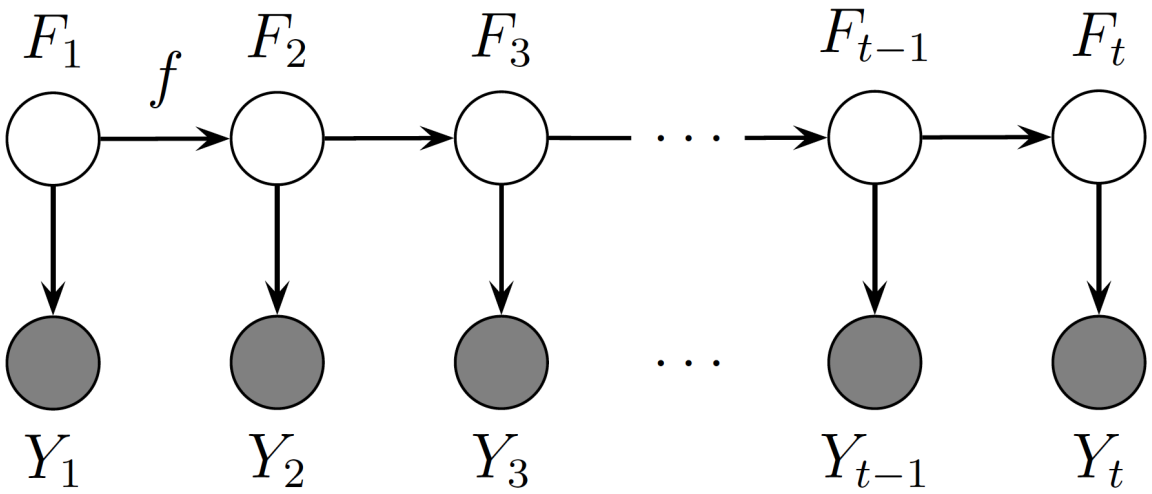


The Kalman filter is a Gaussian process with a special covariance function, one that gives a sparse precision matrix. This allows efficient computation.

$$p(\, f_1 \mid I \,) \triangleq \mathcal{N}(f_1; \nu, \Lambda)$$

$$p(\, f_k \mid f_{k-1},\, I \,) \triangleq \mathcal{N}(f_k; G\, f_{k-1}, Q)$$

$$p(\, y_k \mid f_k,\, I \,) \triangleq \mathcal{N}(y_k; H\, f_k, R)$$



$$p(\, f_1 \mid I \,) \triangleq \mathcal{N}(f_1; \nu, \Lambda)$$

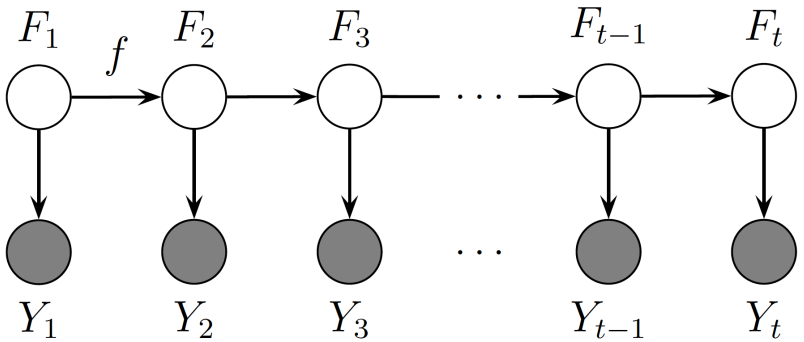
$$p(\, f_k \mid f_{k-1},\, I \,) \triangleq \mathcal{N}(f_k; G\, f_{k-1}, Q)$$

$$p(\, y_k \mid f_k,\, I \,) \triangleq \mathcal{N}(y_k; H\, f_k, R)$$

$$K_f(t_j, t_k) \triangleq G^j\, \Lambda\, G^{\mathsf{T},k} + \sum_{a=1}^{\min(j,k)} G^{j-a}\, Q\, G^{\mathsf{T},k-a}$$

$$K_{f,y}(t_j, t_k) \triangleq K_f(t_j, t_k)\, H^{\mathsf{T}}$$

$$K_y(t_j, t_k) \triangleq H\, K_f(t_j, t_k)\, H^{\mathsf{T}} + \delta_{j,k}\, R$$



$$p(\, f_1 \mid I \,) \triangleq \mathcal{N}(f_1; \nu, \Lambda)$$

$$p(\, f_k \mid f_{k-1},\, I \,) \triangleq \mathcal{N}(f_k; G\, f_{k-1}, Q)$$

$$p(\, y_k \mid f_k,\, I \,) \triangleq \mathcal{N}(y_k; H\, f_k, R)$$

$$K_f(t_j, t_k) \triangleq G^j\, \Lambda\, G^{\mathsf{T},k} + \sum_{a=1}^{\min(j,k)} G^{j-a}\, Q\, G^{\mathsf{T},k-a}$$

$$K_{f,y}(t_j, t_k) \triangleq K_f(t_j, t_k)\, H^{\mathsf{T}}$$

$$K_y(t_j, t_k) \triangleq H\, K_f(t_j, t_k)\, H^{\mathsf{T}} + \delta_{j,k}\, R$$

The Kalman filter's covariance is non-stationary: it grows with time!

Consider tracking a 2D state $\mathbf{x} = (x, y)$.

Example: a feature in an image, or a vehicle position on an approximately planar road or air temperature/air pressure.

Suppose at time t_k the feature or vehicle position has prior

$$p(\mathbf{x}_k \mid \mathbf{z}_{o:k-1}) = \mathcal{N}(\mathbf{x}_k; \boldsymbol{\mu}_{k|k-1}, \boldsymbol{\Sigma}_{k|k-1})$$

$k \mid k-1$ means at timestep k but using only measurements made up to timestep $k-1$. That is, before making a measurement at timestep k. $k \mid k$ is at timestep k including all measurements up to and including timestep k.

Now make a measurement $\mathbf{z}_k$ of $\mathbf{x}_k$ at time k, with

$$p(\mathbf{z}_k \mid \mathbf{x}_k) = \mathcal{N}(\mathbf{z}_k; \mathbf{x}_k, \boldsymbol{R}_k)$$

We are after the posterior density $p(\mathbf{x}_k \mid \mathbf{z}_k)$. Summing precision, we know that the covariance must be updated to

$$\boldsymbol{\Sigma}_{k|k}^{-1} = \boldsymbol{\Sigma}_{k|k-1}^{-1} + \boldsymbol{R}_k^{-1}$$

♣ Example: using priors for tracking.

Prior:

$$\boldsymbol{\mu}_{k|k-1} = \begin{bmatrix} -1 \\ -1 \end{bmatrix} \quad \boldsymbol{\Sigma}_{k|k-1} = \begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix}$$

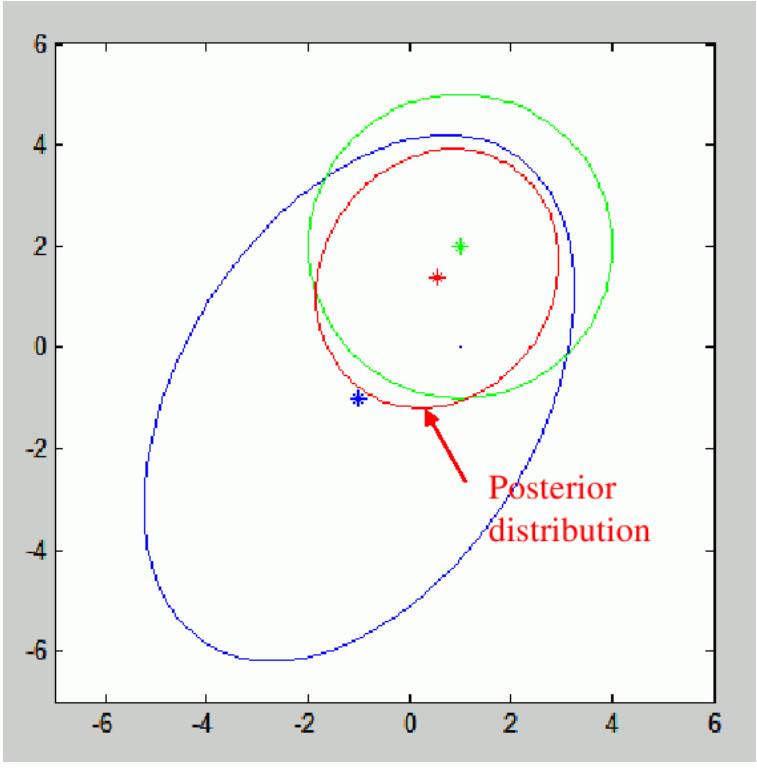
Likelihood:

$$\mathbf{z}_k = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \quad \boldsymbol{R}_k = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Posterior:

$$\boldsymbol{\mu}_{k|k} = \begin{bmatrix} 0.55 \\ 1.37 \end{bmatrix}$$

$$\boldsymbol{\Sigma}_{k|k} = \begin{bmatrix} 0.64 & 0.09 \\ 0.09 & 0.73 \end{bmatrix}$$



The prior is the posterior from the previous time step, updated to take account of the dynamics.

Example: assume that the velocity at timestep k, $\mathbf{u}_k$ is constant, except for noise (i.e. the acceleration is pure noise)

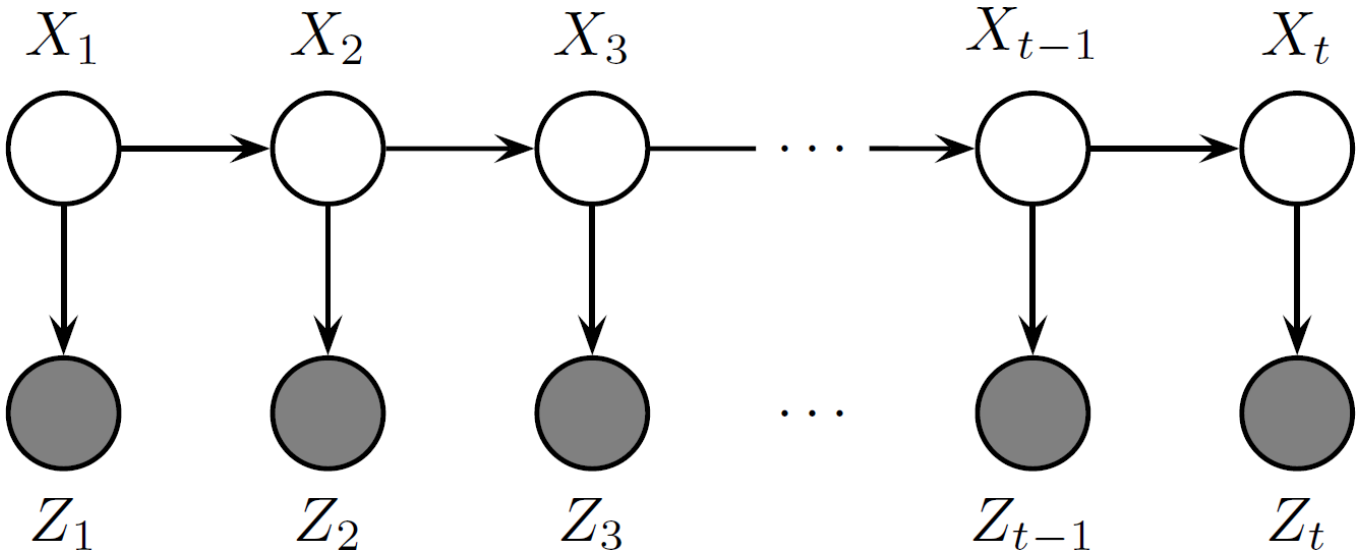
$$\mathbf{x}_k = \mathbf{x}_{k-1} + \mathbf{u}_{k-1}\,\Delta t$$

$$p(\mathbf{u}_{k-1}) = \mathcal{N}\left(\mathbf{u}_{k-1}; \mathbf{v}, \boldsymbol{Q}_k = \sigma^2 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right)$$

where $\Delta t = (t_k - t_{k-1})$: we'll take $\Delta t = 1$ for simplicity hencforth. Note that $\mathbf{x}_k$ is the sum of two Gaussian-distributed variables, $\mathbf{x}_{k-1}$ and $\mathbf{u}_k$. Hence, $\mathbf{x}_k$ is another Gaussian with mean and variance

$$\begin{aligned} \boldsymbol{\mu}_{k|k-1} &= \boldsymbol{\mu}_{k-1|k-1} + \mathbf{v} \\ \boldsymbol{\Sigma}_{k|k-1} &= \boldsymbol{\Sigma}_{k-1|k-1} + \boldsymbol{Q}_k. \end{aligned}$$

The Kalman Filter is representable using a Markov chain.



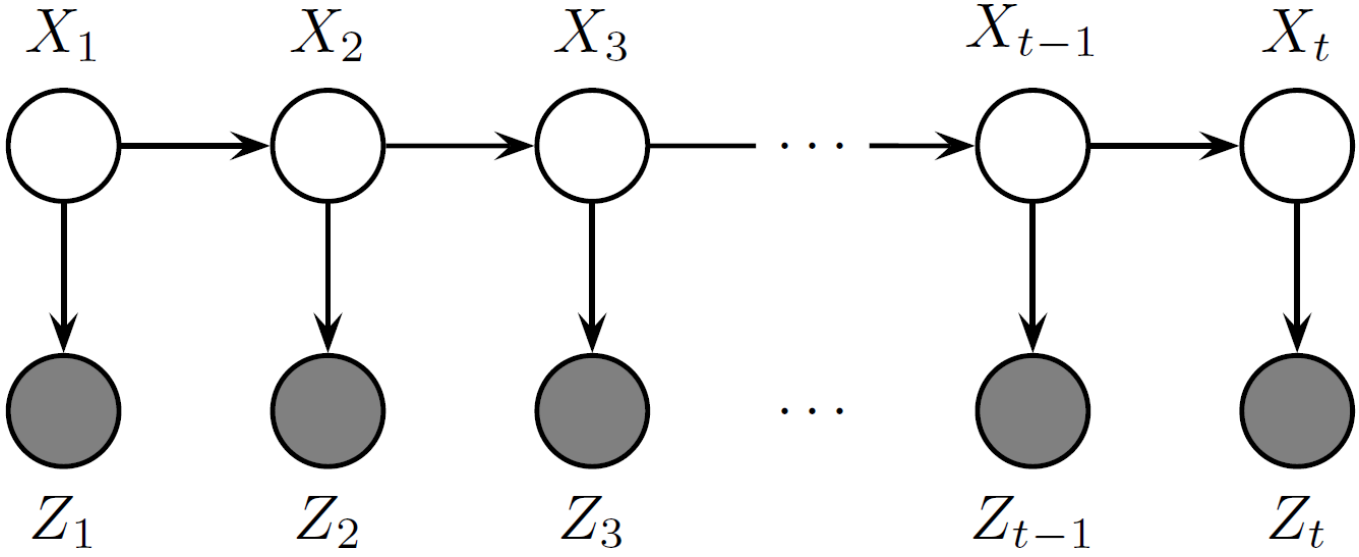
$\mathbf{X}_i$: Unknown state at time-step t_i.

$\mathbf{Z}_i$: Known measurement at time-step t_i.

Observations depend only on the corresponding state.

The states only depend on their neighbours. Given $\mathbf{X}_{i-1}$ and $\mathbf{X}_{i+1}$, $\mathbf{X}_i$ is independent of all other states: the precision matrix over all observations would be sparse.

The Kalman Filter is representable using a Markov chain.



Using the product rule,

$$p(\mathbf{X}_1, \mathbf{X}_2, \mathbf{X}_3, \mathbf{X}_4)$$
$$= p(\mathbf{X}_1)\, p(\mathbf{X}_2 \mid \mathbf{X}_1)\, p(\mathbf{X}_3 \mid \mathbf{X}_2, \mathbf{X}_1)\, p(\mathbf{X}_4 \mid \mathbf{X}_3, \mathbf{X}_2, \mathbf{X}_1)$$

and using the Markov chain

$$= p(\mathbf{X}_1)\, p(\mathbf{X}_2 \mid \mathbf{X}_1)\, p(\mathbf{X}_3 \mid \mathbf{X}_2)\, p(\mathbf{X}_4 \mid \mathbf{X}_3).$$

The Kalman Filter is given by the update cycle:

Posterior after step $k-1$:

$$p(\mathbf{x}_{k-1} \mid \mathbf{z}_{0:k-1}) = \mathcal{N}(\mathbf{x}_{k-1}; \boldsymbol{\mu}_{k-1|k-1}, \boldsymbol{\Sigma}_{k-1|k-1}).$$

Assume $p(\mathbf{x}_k \mid \mathbf{x}_{k-1}) = \mathcal{N}(\mathbf{x}_k; \mathbf{x}_{k-1} + \mathbf{v}, \boldsymbol{Q}_k)$.

1. Prediction: $p(\mathbf{x}_k \mid \mathbf{z}_{0:k-1}) = \mathcal{N}(\mathbf{x}_k; \boldsymbol{\mu}_{k|k-1}, \boldsymbol{\Sigma}_{k|k-1})$, for

$$\begin{aligned} \boldsymbol{\mu}_{k|k-1} &= \boldsymbol{\mu}_{k-1|k-1} + \mathbf{v} \\ \boldsymbol{\Sigma}_{k|k-1} &= \boldsymbol{\Sigma}_{k-1|k-1} + \boldsymbol{Q}_k. \end{aligned}$$

2. Make measurement $\mathbf{z}_k$, $p(\mathbf{z}_k \mid \mathbf{x}_k) = \mathcal{N}(\mathbf{z}_k; \mathbf{x}_k, \boldsymbol{R}_k)$.
3. Fuse measurement with prediction

$$\begin{aligned} \boldsymbol{\Sigma}_{k|k}^{-1} &= \boldsymbol{\Sigma}_{k|k-1}^{-1} + \boldsymbol{R}_k^{-1} \\ \boldsymbol{\mu}_{k|k} &= \boldsymbol{\Sigma}_{k|k}(\boldsymbol{\Sigma}_{k|k-1}^{-1}\boldsymbol{\mu}_{k|k-1} + \boldsymbol{R}_k^{-1}\mathbf{z}_k) \end{aligned}$$

Posterior after step k: $p(\mathbf{x}_k \mid \mathbf{z}_{0:k}) = \mathcal{N}(\mathbf{x}_k; \boldsymbol{\mu}_{k|k}, \boldsymbol{\Sigma}_{k|k})$.

♣ Example.

Posterior $t = 0$	Dynamics step 0 to 1	Measure at $t = 1$
$\mathbf{x}_{0\|0} = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$	$\mathbf{v} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$	$\mathbf{z}_1 = \begin{bmatrix} 6 \\ 2 \end{bmatrix}$
$\Sigma_{0\|0} = \begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix}$	$\boldsymbol{Q}_1 = \begin{bmatrix} 0.3 & 0 \\ 0 & 0.3 \end{bmatrix}$	$\boldsymbol{R}_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

Prediction at $t = 1$:

$$\Sigma_{1|0} = \Sigma_{0|0} + \boldsymbol{Q}_1 = \begin{bmatrix} 2.3 & 1.0 \\ 1.0 & 3.3 \end{bmatrix} \qquad \mathbf{x}_{1|0} = \mathbf{x}_{0|0} + \mathbf{v} = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$$

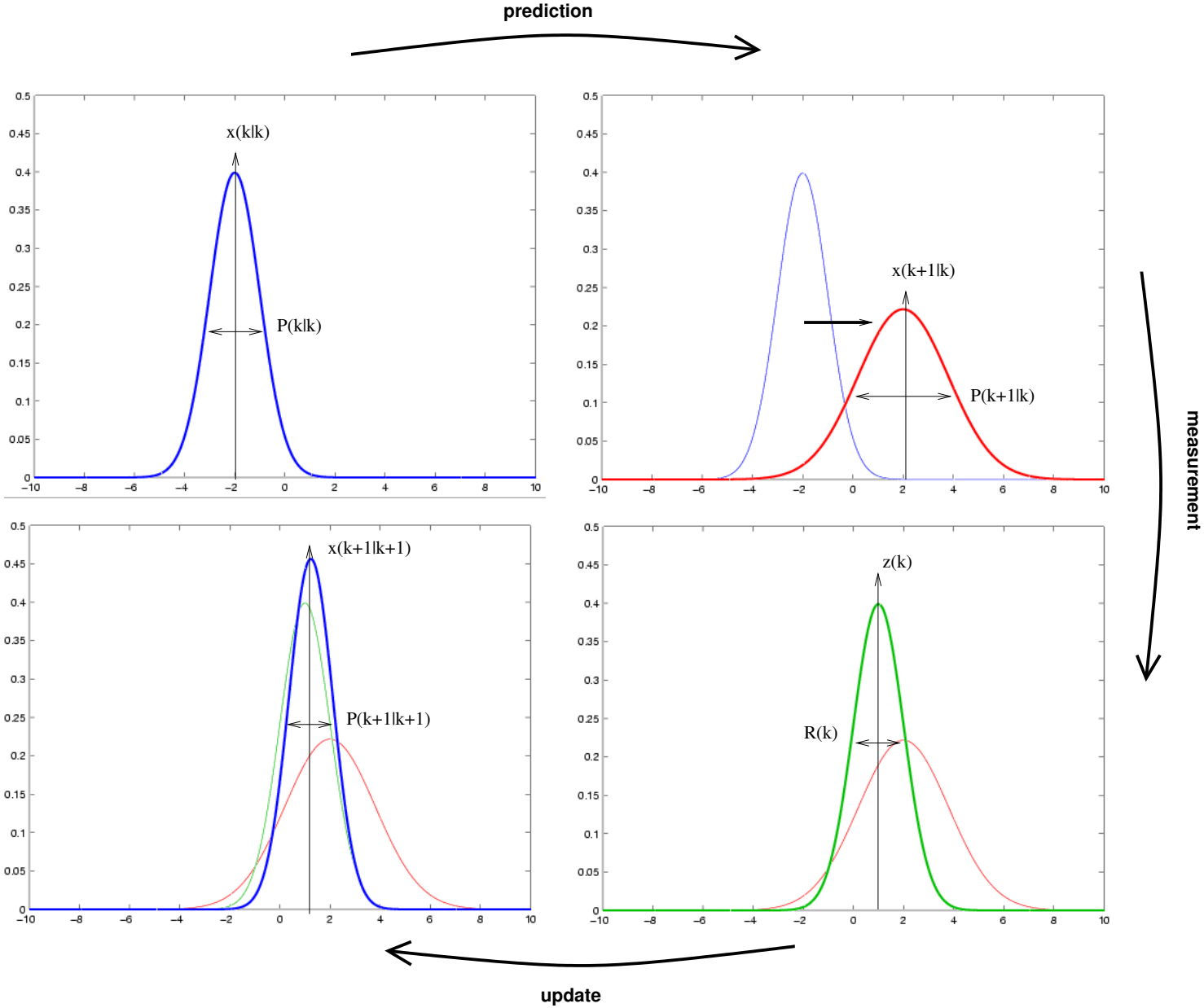
Fuse measurement at $t = 1$:

$$\Sigma_{1|1} = \left(\Sigma_{1|0}^{-1} + \boldsymbol{R}_1^{-1}\right)^{-1} = \begin{bmatrix} 0.674 & 0.076 \\ 0.076 & 0.750 \end{bmatrix}$$

$$\mathbf{x}_{1|1} = \Sigma_{1|1}\left(\Sigma_{1|0}^{-1}\,\mathbf{x}_{1|0} + \boldsymbol{R}_1^{-1}\,\mathbf{z}_1\right) = \begin{bmatrix} 3.95 \\ 1.78 \end{bmatrix}$$

Berkeley Tracker in action

Graphically ...



A richer Kalman Filter:

Posterior after step $k-1$:

$p(\mathbf{x}_{k-1} \mid \mathbf{z}_{0:k-1}) = \mathcal{N}(\mathbf{x}_{k-1}; \boldsymbol{\mu}_{k-1|k-1}, \boldsymbol{\Sigma}_{k-1|k-1})$.

Assume $p(\mathbf{x}_k \mid \mathbf{x}_{k-1}) = \mathcal{N}(\mathbf{x}_k; \boldsymbol{F}\mathbf{x}_{k-1}, \boldsymbol{Q}_k)$.

1. Prediction: $p(\mathbf{x}_k \mid \mathbf{z}_{0:k-1}) = \mathcal{N}(\mathbf{x}_k; \boldsymbol{\mu}_{k|k-1}, \boldsymbol{\Sigma}_{k|k-1})$, for

$$\begin{aligned} \boldsymbol{\mu}_{k|k-1} &= \boldsymbol{F}\boldsymbol{\mu}_{k-1|k-1} \\ \boldsymbol{\Sigma}_{k|k-1} &= \boldsymbol{F}\boldsymbol{\Sigma}_{k-1|k-1}\boldsymbol{F}^\top + \boldsymbol{Q}_k. \end{aligned}$$

2. Make measurement $\mathbf{z}_k$, $p(\mathbf{z}_k \mid \mathbf{x}_k) = \mathcal{N}(\mathbf{z}_k; \boldsymbol{H}\mathbf{x}_k, \boldsymbol{R}_k)$.
3. Fuse measurement with prediction

$$\begin{aligned} \boldsymbol{\Sigma}_{k|k}^{-1} &= \boldsymbol{\Sigma}_{k|k-1}^{-1} + \boldsymbol{H}^\top \boldsymbol{R}_k^{-1}\boldsymbol{H} \\ \boldsymbol{\mu}_{k|k} &= \boldsymbol{\Sigma}_{k|k}(\boldsymbol{\Sigma}_{k|k-1}^{-1}\boldsymbol{\mu}_{k|k-1} + \boldsymbol{H}^\top \boldsymbol{R}_k^{-1}\mathbf{z}_k) \end{aligned}$$

Posterior after step k: $p(\mathbf{x}_k \mid \mathbf{z}_{0:k}) = \mathcal{N}(\mathbf{x}_k; \boldsymbol{\mu}_{k|k}, \boldsymbol{\Sigma}_{k|k})$.

Let's assume our state evolves linearly (according to $\boldsymbol{F}$) over a time-step.

We also regard $\mathbf{v}$ as a member of the state $\mathbf{x} = \left[x, y, v_x, v_y\right]^\top$,

$$\text{e.g. } \mathbf{x}_k = \begin{bmatrix} x \\ y \\ v_x \\ v_y \end{bmatrix}_k = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ v_x \\ v_y \end{bmatrix}_{k-1} + \varepsilon_{k-1},$$

$$p(\varepsilon_{k-1}) = \mathcal{N}\left(\varepsilon_{k-1}; \mathbf{0}_{4\times 1}, \boldsymbol{Q}_k\right)$$

But recall that if $\mathbf{y} = \boldsymbol{A}\,\mathbf{x}$, then $\Sigma_y = \boldsymbol{A}\,\Sigma_x\,\boldsymbol{A}^\top$.

Using this explains the prediction phase:

State	$\boldsymbol{\mu}_{k\|k-1} = \boldsymbol{F}\,\boldsymbol{\mu}_{k-1\|k-1}$
Covariance	$\Sigma_{k\|k-1} = \boldsymbol{F}\,\Sigma_{k-1\|k-1}\,\boldsymbol{F}^\top + \boldsymbol{Q}_k$

We assume measurements are linearly related to the state.

That is, we have $p(\mathbf{z}_k \mid \mathbf{x}_k) = \mathcal{N}(\mathbf{z}_k; \boldsymbol{H}\,\mathbf{x}_k, \boldsymbol{R}_k)$. Here we will assume that we can only measure x and y, not velocities, so

$$\mathbf{z}_k = \boldsymbol{H}\,\mathbf{x}_k + \boldsymbol{\eta}_k = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ v_x \\ v_y \end{bmatrix}_k + \boldsymbol{\eta}_k$$

$$p(\boldsymbol{\eta}_k) = \mathcal{N}\left(\boldsymbol{\eta}_k; \mathbf{0}_{4\times 1}, \boldsymbol{R}_k\right)$$

The Woodbury matrix identity

$$(\boldsymbol{A} + \boldsymbol{U}\,\boldsymbol{C}\,\boldsymbol{V})^{-1} = \boldsymbol{A} - \boldsymbol{A}^{-1}\,\boldsymbol{U}(\boldsymbol{C}^{-1} + \boldsymbol{V}\boldsymbol{A}^{-1}\boldsymbol{U})^{-1}\boldsymbol{V}\,\boldsymbol{A}^{-1}$$

along with Gaussian identities gives the update phase:

State	$\boldsymbol{\mu}_{k\mid k} = \boldsymbol{\Sigma}_{k\mid k}(\boldsymbol{\Sigma}^{-1}_{k\mid k-1}\boldsymbol{\mu}_{k\mid k-1} + \boldsymbol{H}^\top \boldsymbol{R}_k^{-1}\mathbf{z}_k)$
Covariance	$\boldsymbol{\Sigma}_{k\mid k} = \left(\boldsymbol{\Sigma}^{-1}_{k\mid k-1} + \boldsymbol{H}^\top \boldsymbol{R}_k^{-1}\boldsymbol{H}\right)^{-1}$

Bayes' Theorem gives a more satisfying derivation of the update phase.

Bayes' theorem says that the posterior probability of the state $\mathbf{x}$ given the data $\mathbf{z}$ is $p(\mathbf{x} \mid \mathbf{z}) = p(\mathbf{z} \mid \mathbf{x})p(\mathbf{x})/p(\mathbf{z})$. The prior of the data is a constant, and of no interest.

Assume that the prior of the state (we're not writing the conditioning on $\mathbf{z}_{0:k-1}$ for convenience) is a multivariate Gaussian centred on $\boldsymbol{\mu}_{k|k-1}$ with covariance $\boldsymbol{\Sigma}_{k|k-1}$, then

$$p(\mathbf{x}_k) \propto \exp\left(-\tfrac{1}{2}\left(\mathbf{x}_k - \boldsymbol{\mu}_{k|k-1}\right)^\top \boldsymbol{\Sigma}_{k|k-1}^{-1}\left(\mathbf{x}_k - \boldsymbol{\mu}_{k|k-1}\right)\right).$$

Similarly

$$p(\mathbf{z}_k \mid \mathbf{x}_k) \propto \exp\left(-\tfrac{1}{2}\left(\mathbf{z}_k - \boldsymbol{H}\,\mathbf{x}_k\right)^\top \boldsymbol{R}_k^{-1}\left(\mathbf{z}_k - \boldsymbol{H}\,\mathbf{x}_k\right)\right).$$

We'll define $p(\mathbf{x}_k \mid \mathbf{z}_k)$ as having the form

$$p(\mathbf{x}_k \mid \mathbf{z}_k) \propto \exp\left(-\tfrac{1}{2}\left(\mathbf{x}_k - \boldsymbol{\mu}_{k|k}\right)^\top \boldsymbol{\Sigma}_{k|k}^{-1}\left(\mathbf{x}_k - \boldsymbol{\mu}_{k|k}\right)\right).$$

Now we take the natural log of Bayes' rule.

$$\begin{aligned} -\ln p(\mathbf{x}_k \mid \mathbf{z}_k) &= \tfrac{1}{2}\left(\mathbf{x}_k - \boldsymbol{\mu}_{k|k}\right)^\top \boldsymbol{\Sigma}_{k|k}^{-1}\left(\mathbf{x}_k - \boldsymbol{\mu}_{k|k}\right) + c_1 \\ &= \tfrac{1}{2}\left(\mathbf{z}_k - \boldsymbol{H}\,\mathbf{x}_k\right)^\top \boldsymbol{R}_k^{-1}\left(\mathbf{z}_k - \boldsymbol{H}\,\mathbf{x}_k\right) \\ &\quad + \tfrac{1}{2}\left(\mathbf{x}_k - \boldsymbol{\mu}_{k|k-1}\right)^\top \boldsymbol{\Sigma}_{k|k-1}^{-1}\left(\mathbf{x}_k - \boldsymbol{\mu}_{k|k-1}\right) + c_2 \end{aligned}$$

If you now complete the square by gathering terms in $\mathbf{x}_k$ (that is, $\mathbf{x}_k^\top \boldsymbol{A}\mathbf{x}_k$ and $\boldsymbol{B}\mathbf{x}_k$), you'll find

$$-\ln p(\mathbf{x}_k \mid \mathbf{z}_k) = \tfrac{1}{2}\left(\mathbf{x}_k - \boldsymbol{\mu}_{k|k}\right)^\top \boldsymbol{\Sigma}_{k|k}^{-1}\left(\mathbf{x}_k - \boldsymbol{\mu}_{k|k}\right)$$

where

$$\begin{aligned} \boldsymbol{\mu}_{k|k} &= \boldsymbol{\Sigma}_{k|k}\left(\boldsymbol{\Sigma}_{k|k-1}^{-1}\boldsymbol{\mu}_{k|k-1} + \boldsymbol{H}^\top \boldsymbol{R}_k^{-1}\mathbf{z}_k\right) \\ \boldsymbol{\Sigma}_{k|k} &= \left(\boldsymbol{\Sigma}_{k|k-1}^{-1} + \boldsymbol{H}^\top \boldsymbol{R}_k^{-1}\boldsymbol{H}\right)^{-1} \end{aligned}$$

The Kalman Filter can be extended to manage the *n*th-order autoregressive model.

We often want to allow the next position to be a function of not just the last position, but the last n positions. To do so, we just augment the state with those extra positions. If $n = 2$, we have the state $\mathbf{x}_k = [x_k, x_{k-1}]^\top$,

$$\text{e.g. } \mathbf{x}_k = \begin{bmatrix} x_k \\ x_{k-1} \end{bmatrix} = \begin{bmatrix} \alpha & \beta \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x_{k-1} \\ x_{k-2} \end{bmatrix} + \varepsilon_{k-1},$$

$$p(\varepsilon_{k-1}) = \mathcal{N}(\varepsilon_{k-1}; \mathbf{0}_{2\times 1}, \boldsymbol{Q}_k)$$

The Kalman Filter can then be trivially applied with

$$\boldsymbol{F} = \begin{bmatrix} \alpha & \beta \\ 1 & 0 \end{bmatrix}$$

The Kalman Filter is an optimal recursive filter for systems where

- the initial state is described by a Gaussian distribution,
- the measurement noise and prediction or "plant" noise are Gaussian, and
- where both the state evolution is linear ($\boldsymbol{F}$) and the measurements are linearly related to the state ($\boldsymbol{H}$).

We now briefly mention two different approaches to filtering:

- The Extended Kalman Filter, useful when the state evolution and/or the measurement model is non-linear, but the distributions are Gaussian.
- The Particle Filter, useful when the distributions are very non-Gaussian.

The Extended Kalman Filter begins with a Gaussian "prior".

We begin with $p(\mathbf{x}_{k-1}) = \mathcal{N}(\mathbf{x}_{k-1}; \boldsymbol{\mu}_{k-1|k-1}, \Sigma_{k-1|k-1})$.

However, rather than evolving linearly as $\mathbf{x}_k = \boldsymbol{F}\,\mathbf{x}_{k-1}$ the state may evolve non-linearly as

$$\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}).$$

Now assume that the non-linear vector function $\mathbf{f}(\cdot)$ is close to linear, so the propagation of uncertainty can be linearized about the current operating point.

Recall the Jacobian matrix.

To avoid too many $\mathbf{x}$'s and too many subscripts, suppose $\mathbf{y} = \mathbf{f}(\mathbf{x})$.
$\mathbf{f}$ is a "vector function" — that is a vector of functions. For example, suppose $\mathbf{y}$ and $\mathbf{x}$ have length 3.

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} f_1(x_1, x_2, x_3) \\ f_2(x_1, x_2, x_3) \\ f_3(x_1, x_2, x_3) \end{bmatrix} = \begin{bmatrix} x_1^2 + 3x_2 + 1/x_3 \\ -1/x_1 + \cos(x_3) \\ x_1 + x_2/x_3 + x_3 \end{bmatrix}$$

Then

$$\nabla \mathbf{f} = \begin{bmatrix} \partial y_1/\partial x_1 & \partial y_1/\partial x_2 & \partial y_1/\partial x_3 \\ \partial y_2/\partial x_1 & \partial y_2/\partial x_2 & \partial y_2/\partial x_3 \\ \partial y_3/\partial x_1 & \partial y_3/\partial x_2 & \partial y_3/\partial x_3 \end{bmatrix} = \begin{bmatrix} 2x_1 & 3 & -1/x_3^2 \\ 1/x_1^2 & 0 & -\sin(x_3) \\ 1 & 1/x_3 & -x_2/x_3^2 + 1 \end{bmatrix}$$

The numerical values in $\nabla \mathbf{f}$ are computed at the current value of $\mathbf{x}$.

Suppose we have the linear relation $\mathbf{y} = \boldsymbol{A}\,\mathbf{x}$.

Now rewrite $\mathbf{y}$ as a slight deviation from $\mathbf{y}_o = \boldsymbol{A}\,\mathbf{x}_o$,

$$\mathbf{y} = \mathbf{y}_o + \boldsymbol{\delta}\mathbf{y} = \boldsymbol{A}\,\mathbf{x} = \boldsymbol{A}\,(\mathbf{x}_o + \boldsymbol{\delta}\mathbf{x}) \quad \Rightarrow \quad \boldsymbol{\delta}\mathbf{y} = \boldsymbol{A}\,\boldsymbol{\delta}\mathbf{x}$$

Now, no matter what the fixed point, the covariance of $\mathbf{y}$ is

$$\boldsymbol{\Sigma}_y = \mathbb{E}[\boldsymbol{\delta}\mathbf{y}\boldsymbol{\delta}\mathbf{y}^\top] = \mathbb{E}[\boldsymbol{A}\boldsymbol{\delta}\mathbf{x}\boldsymbol{\delta}\mathbf{x}^\top\boldsymbol{A}^\top] = \boldsymbol{A}\mathbb{E}[\boldsymbol{\delta}\mathbf{x}\boldsymbol{\delta}\mathbf{x}^\top]\boldsymbol{A}^\top = \boldsymbol{A}\boldsymbol{\Sigma}_x\boldsymbol{A}^\top$$

.

Now suppose $\mathbf{y} = \mathbf{f}(\mathbf{x})$. A 1st order Taylor expansion gives

$$\mathbf{y} = \mathbf{y}_o + \boldsymbol{\delta}\mathbf{y} = \mathbf{f}(\mathbf{x}) \approx \mathbf{f}(\mathbf{x}_o) + \frac{\partial \mathbf{f}}{\partial \mathbf{x}}\boldsymbol{\delta}\mathbf{x} \quad \Rightarrow \quad \boldsymbol{\delta}\mathbf{y} = \frac{\partial \mathbf{f}}{\partial \mathbf{x}}\boldsymbol{\delta}\mathbf{x} = \nabla\mathbf{f}\,\boldsymbol{\delta}\mathbf{x}$$

$$\boldsymbol{\Sigma}_y = \nabla\mathbf{f}\,\boldsymbol{\Sigma}_x\nabla\mathbf{f}^\top$$

Similarly, measurements can be non-linearly related to the state.

Rather than $\mathbf{z} = \boldsymbol{H}\,\mathbf{x}$ we assume

$$\mathbf{z} = \mathbf{h}(\mathbf{x})$$

And just as $\boldsymbol{F}$ was replaced by $\nabla\mathbf{f}$, $\boldsymbol{H}$ is replaced by $\nabla\mathbf{h}$, where

$$\nabla\mathbf{h} = \begin{bmatrix} \partial z_1/\partial x_1 & \partial z_1/\partial x_2 & \cdots \\ \partial z_2/\partial x_1 & \partial z_2/\partial x_2 & \cdots \\ \vdots & \vdots & \ddots \end{bmatrix}$$

As before, numerical values of $\nabla\mathbf{h}$ are computed at the current value of $\mathbf{x}$.

The Extended Kalman Filter:

Posterior after step $k-1$:

$p(\mathbf{x}_{k-1} \mid \mathbf{z}_{0:k-1}) = \mathcal{N}(\mathbf{x}_{k-1}; \boldsymbol{\mu}_{k-1|k-1}, \boldsymbol{\Sigma}_{k-1|k-1})$.

Assume $p(\mathbf{x}_k \mid \mathbf{x}_{k-1}) = \mathcal{N}(\mathbf{x}_k; \mathbf{f}(\mathbf{x}_{k-1}), \boldsymbol{Q}_k)$.

1. Prediction: $p(\mathbf{x}_k \mid \mathbf{z}_{0:k-1}) = \mathcal{N}(\mathbf{x}_k; \boldsymbol{\mu}_{k|k-1}, \boldsymbol{\Sigma}_{k|k-1})$, for

$$\begin{aligned} \boldsymbol{\mu}_{k|k-1} &= \mathbf{f}(\boldsymbol{\mu}_{k-1|k-1}) \quad \text{NB: we can use } f \text{ exactly!} \\ \boldsymbol{\Sigma}_{k|k-1} &= \nabla\mathbf{f}\, \boldsymbol{\Sigma}_{k-1|k-1}\, \nabla\mathbf{f}^\top + \boldsymbol{Q}_k. \end{aligned}$$

2. Make measurement $\mathbf{z}_k$, $p(\mathbf{z}_k \mid \mathbf{x}_k) = \mathcal{N}(\mathbf{z}_k; \mathbf{h}(\mathbf{x}_k), \boldsymbol{R}_k)$.

3. Fuse measurement with prediction

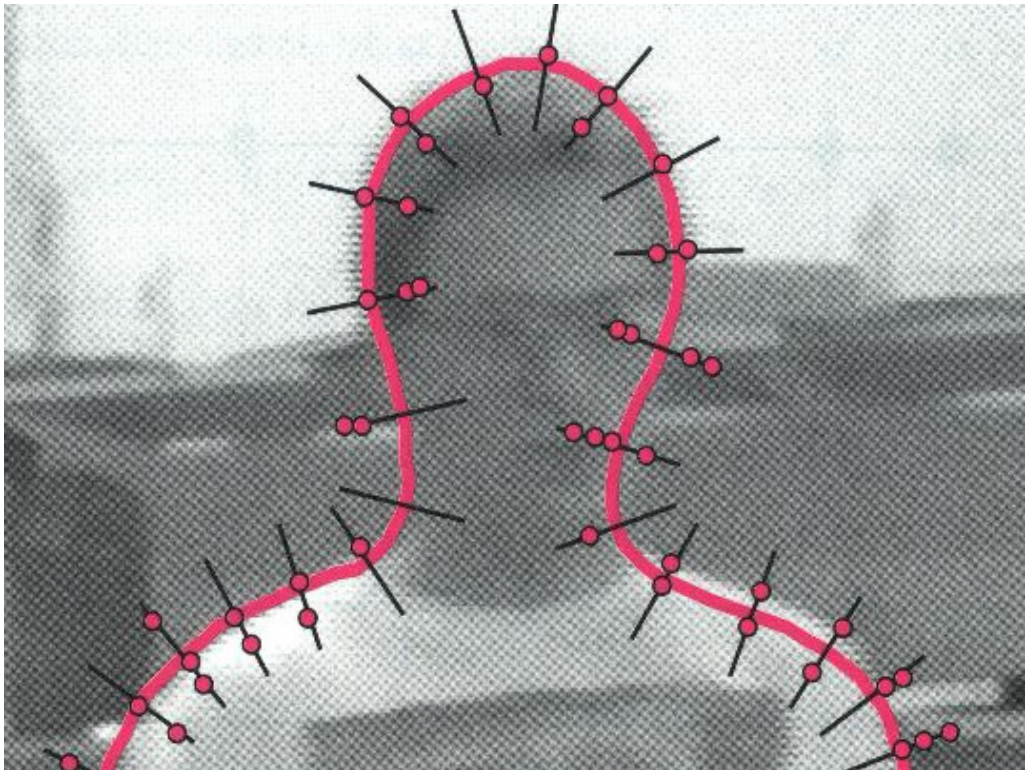
$$\begin{aligned} \boldsymbol{\Sigma}_{k|k}^{-1} &= \boldsymbol{\Sigma}_{k|k-1}^{-1} + \nabla\mathbf{h}^\top \boldsymbol{R}_k^{-1} \nabla\mathbf{h} \\ \boldsymbol{\mu}_{k|k} &= \boldsymbol{\Sigma}_{k|k}(\boldsymbol{\Sigma}_{k|k-1}^{-1}\boldsymbol{\mu}_{k|k-1} + \nabla\mathbf{h}^\top \boldsymbol{R}_k^{-1}\mathbf{z}_k) \end{aligned}$$

Posterior after step k: $p(\mathbf{x}_k \mid \mathbf{z}_{0:k}) = \mathcal{N}(\mathbf{x}_k; \boldsymbol{\mu}_{k|k}, \boldsymbol{\Sigma}_{k|k})$.

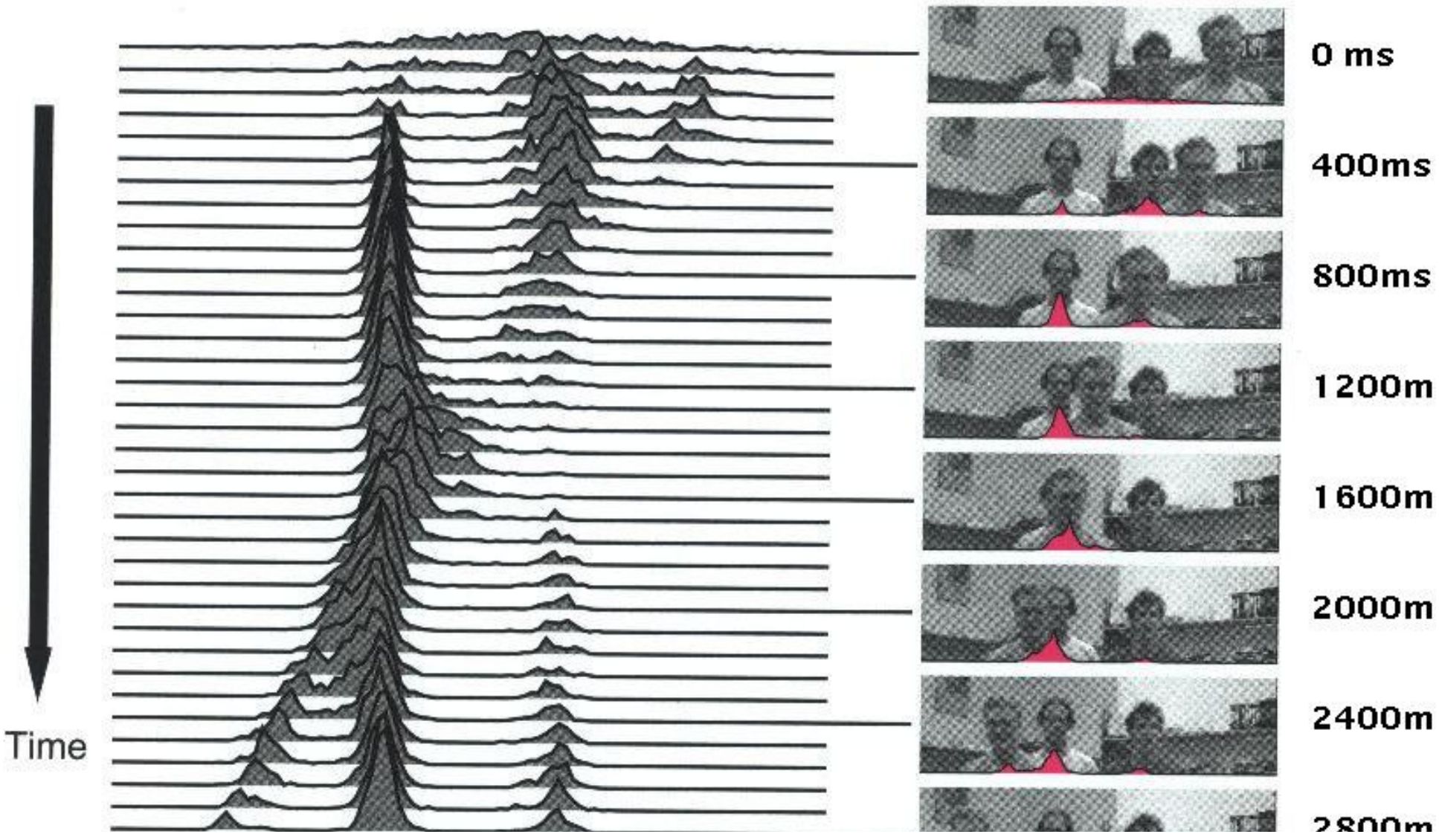
Particle filters are for multimodal densities.

The Kalman Filter can (and usually does) fail catastrophically if the measurement density is multimodal.
Multimodal measurement pdfs enter when we use robust statistical methods to identify rogue data or "outliers".
Multimodal densities also emerge when we wish to represent multiple hypotheses (one mode per hypothesis).

Consider a contour tracker.



Multiple hypotheses give multiple posterior peaks.

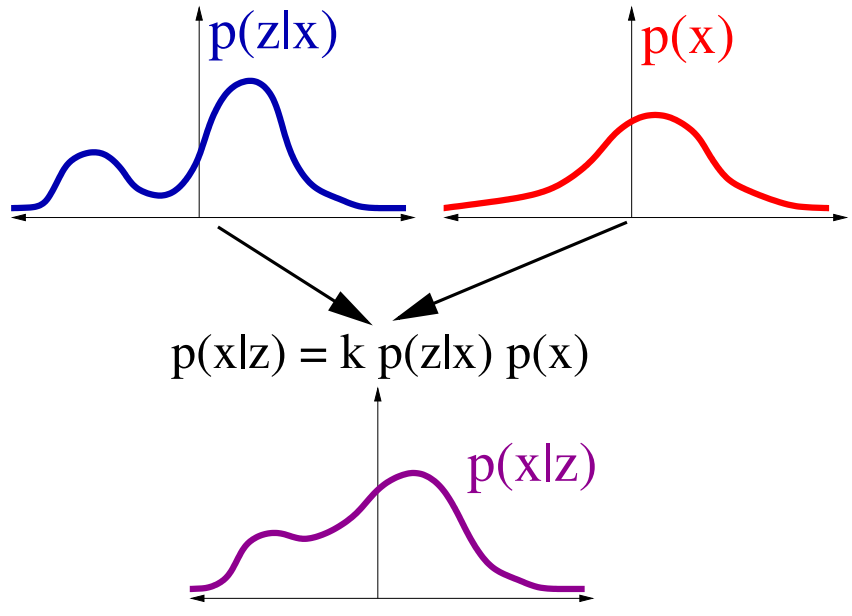


The particle filter is a sample-based means of sequential integration.

Our goal is to compute the posterior using Bayes' rule,

$$p(\mathbf{x} \mid \mathbf{z}) = \frac{p(\mathbf{z} \mid \mathbf{x})\, p(\mathbf{x})}{\int p(\mathbf{z} \mid \mathbf{x})\, p(\mathbf{x})\, \mathrm{d}\mathbf{x}}.$$

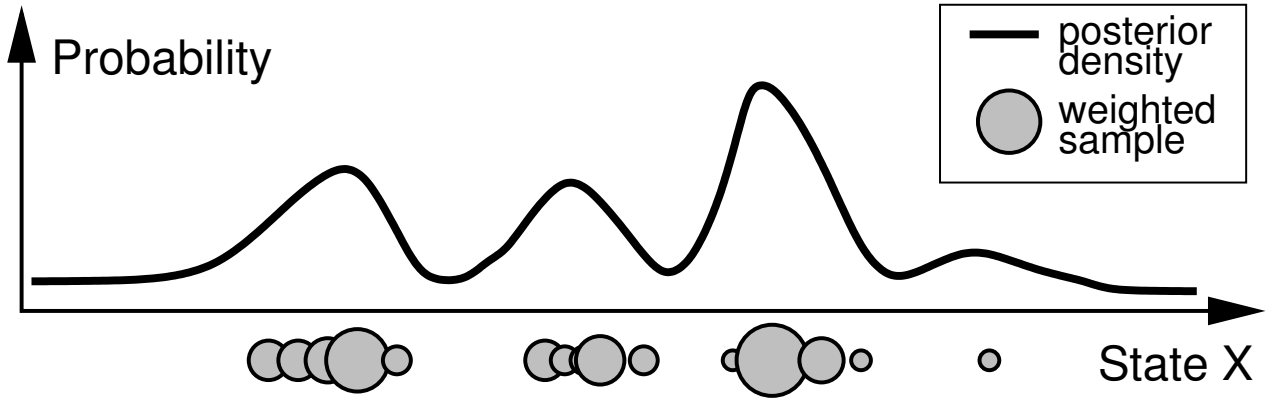
For most multimodal distributions, the integral in the denominator is intractable. Similarly, the integrals for the mean, $\int \mathbf{z}\, p(\mathbf{z} \mid \mathbf{x})\, \mathrm{d}\mathbf{z}$, and covariance are often intractable.



At timestep $k-1$, the posterior density is represented using a set of N weighted particles, $\{(\mathbf{s}_{k-1}^{(i)}, \pi_{k-1}^{(i)}) \mid i = 1 \ldots N\}$.

Each particle represents a particular instance of the state $\mathbf{X}_{k-1} = \mathbf{s}_{k-1}$ at time $k-1$, with a weight π_{k-1} and where $\sum_i \pi_{k-1}^{(i)} = 1$.

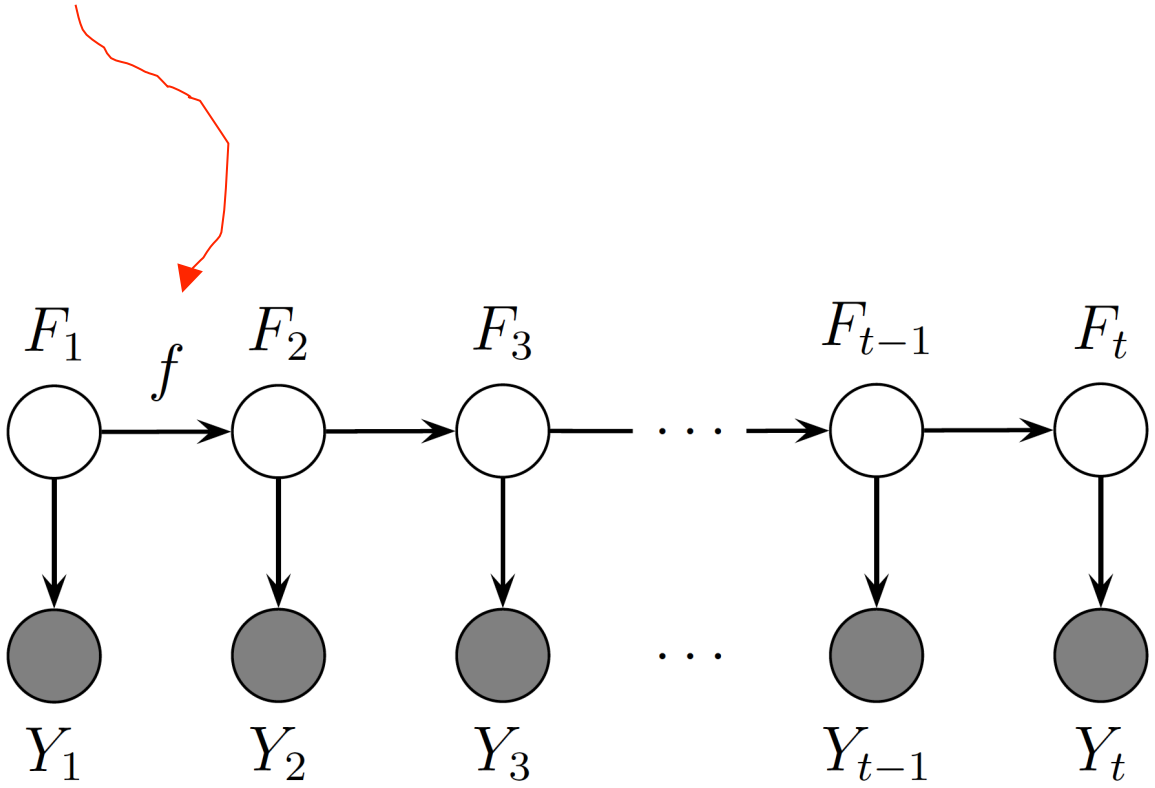
Particles should (i) be clustered and (ii) have high weights where there is high density, it'd be pointless to have lots of particles with $\pi = 0$.



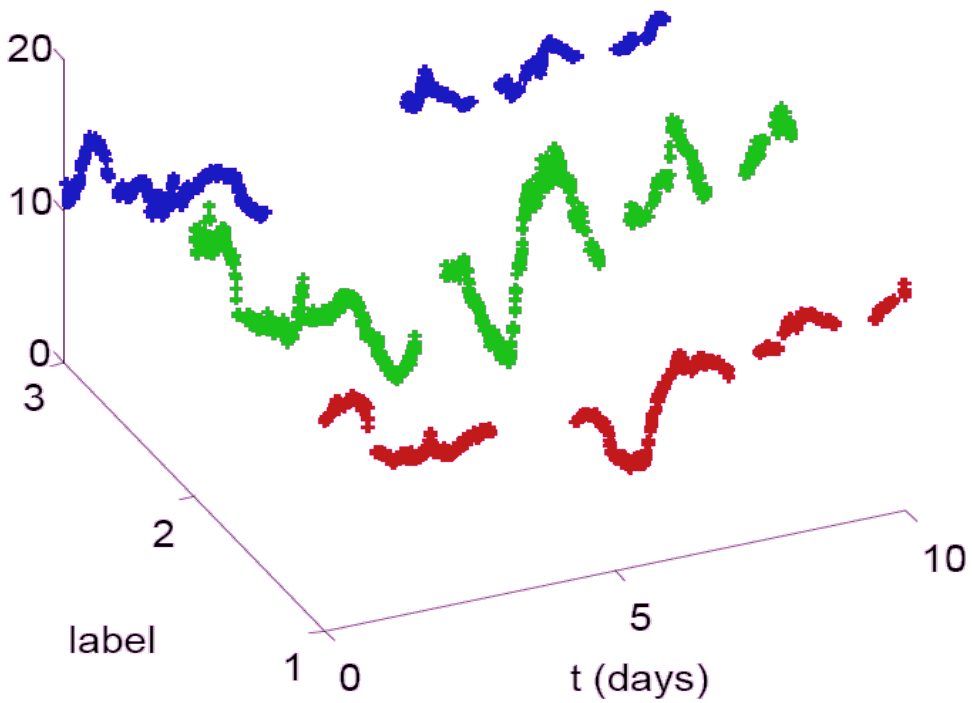
In summary,

1. The Kalman Filter requires Gaussianity: linear measurements and dynamics.
2. The Extended Kalman Filter linearises non-linear measurements and dynamics in order to effect analytic filtering.
3. The Particle Filter uses samples to numerically resolve inference for non-Gaussian systems.

An extension to the Extended Kalman Filter is to use a Gaussian process to model the transition function.



If there are multiple time-series (outputs), reframe the problem as having a single output, and an additional label input specifying the output.



Hence we do not need simultaneous observations of all outputs.

If the inputs were previously x, and outputs were labelled by $l = 1, \ldots, L$, we now need to specify a covariance over both x and l, e.g.

separable for convenience

$$K\left(\left(x_i, l_i\right), \left(x_j, l_j\right)\right) = K\left(x_i, x_j\right) K\left(l_i, l_j\right)$$

If L is not too large, we could use the spherical parameterisation.

We can represent any covariance K using the spherical parameterisation.

$$K = R^{\mathrm{T}} R$$

$$R = \begin{pmatrix} 1 & \cos(\theta_1) & \cos(\theta_2) & \cdots \\ 0 & \sin(\theta_1) & \sin(\theta_2)\cos(\theta_3) & \\ 0 & 0 & \sin(\theta_2)\sin(\theta_3) & \\ \vdots & & & \ddots \end{pmatrix} \begin{pmatrix} h_1 & 0 & 0 & \cdots \\ 0 & h_2 & 0 & \\ 0 & 0 & h_3 & \\ \vdots & & & \ddots \end{pmatrix}$$

Some special large matrices can be represented in a compact way using the Kronecker product.

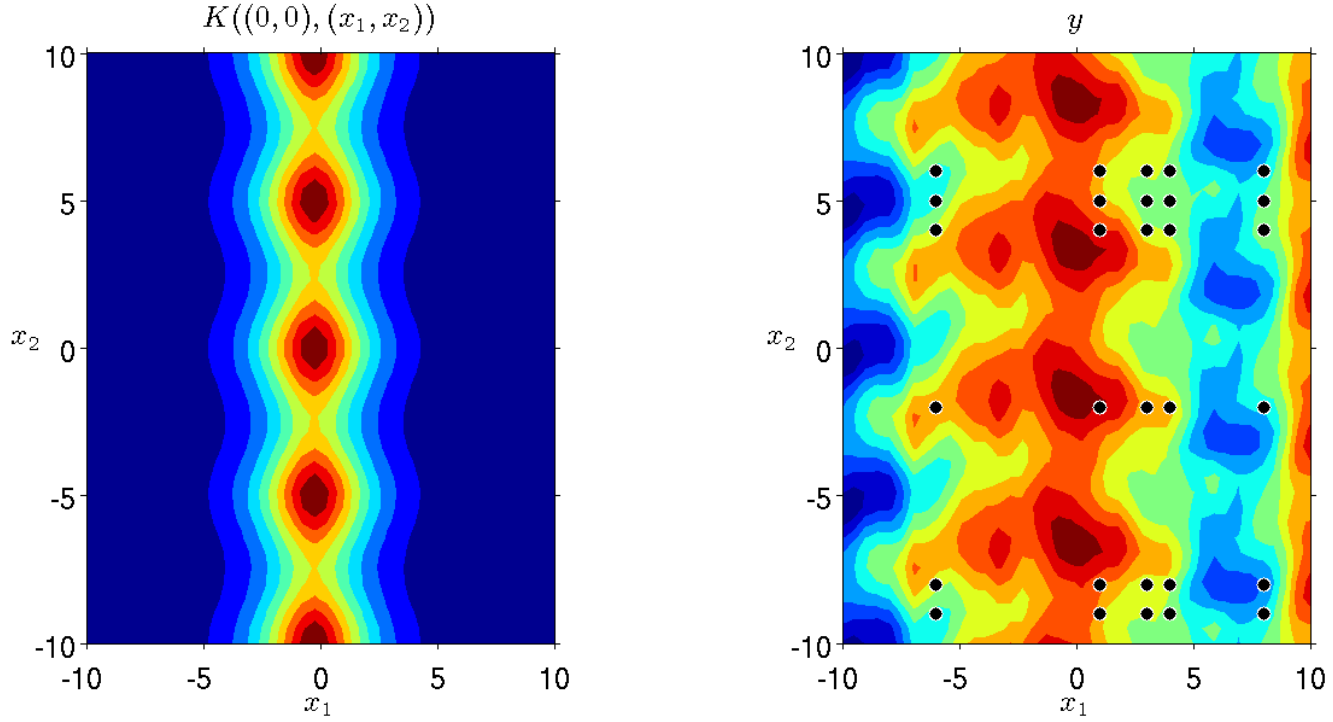
$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}B & \cdots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \cdots & a_{mn}B \end{bmatrix}.$$

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}b_{11} & a_{11}b_{12} & \cdots & a_{11}b_{1q} & \cdots & \cdots & a_{1n}b_{11} & a_{1n}b_{12} & \cdots & a_{1n}b_{1q} \\ a_{11}b_{21} & a_{11}b_{22} & \cdots & a_{11}b_{2q} & \cdots & \cdots & a_{1n}b_{21} & a_{1n}b_{22} & \cdots & a_{1n}b_{2q} \\ \vdots & \vdots & \ddots & \vdots & & & \vdots & \vdots & \ddots & \vdots \\ a_{11}b_{p1} & a_{11}b_{p2} & \cdots & a_{11}b_{pq} & \cdots & \cdots & a_{1n}b_{p1} & a_{1n}b_{p2} & \cdots & a_{1n}b_{pq} \\ \vdots & \vdots & & \vdots & \ddots & & \vdots & \vdots & & \vdots \\ \vdots & \vdots & & \vdots & & \ddots & \vdots & \vdots & & \vdots \\ a_{m1}b_{11} & a_{m1}b_{12} & \cdots & a_{m1}b_{1q} & \cdots & \cdots & a_{mn}b_{11} & a_{mn}b_{12} & \cdots & a_{mn}b_{1q} \\ a_{m1}b_{21} & a_{m1}b_{22} & \cdots & a_{m1}b_{2q} & \cdots & \cdots & a_{mn}b_{21} & a_{mn}b_{22} & \cdots & a_{mn}b_{2q} \\ \vdots & \vdots & \ddots & \vdots & & & \vdots & \vdots & \ddots & \vdots \\ a_{m1}b_{p1} & a_{m1}b_{p2} & \cdots & a_{m1}b_{pq} & \cdots & \cdots & a_{mn}b_{p1} & a_{mn}b_{p2} & \cdots & a_{mn}b_{pq} \end{bmatrix}.$$

e.g.

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \otimes \begin{bmatrix} 0 & 5 \\ 6 & 7 \end{bmatrix} = \begin{bmatrix} 1\cdot 0 & 1\cdot 5 & 2\cdot 0 & 2\cdot 5 \\ 1\cdot 6 & 1\cdot 7 & 2\cdot 6 & 2\cdot 7 \\ 3\cdot 0 & 3\cdot 5 & 4\cdot 0 & 4\cdot 5 \\ 3\cdot 6 & 3\cdot 7 & 4\cdot 6 & 4\cdot 7 \end{bmatrix} = \begin{bmatrix} 0 & 5 & 0 & 10 \\ 6 & 7 & 12 & 14 \\ 0 & 15 & 0 & 20 \\ 18 & 21 & 24 & 28 \end{bmatrix}.$$

A Gaussian process will have a Kronecker product for a covariance matrix if we use a product covariance function and a grid of samples.



For example, taking the covariance that is separable over *t* and *l*, along with simultaneous observations for all sensors, gives a Kronecker product.

UNIVERSITY OF OXFORD

If K is a Kronecker product, there exists a very efficient method to solve $v = K\,x$ for x (particularly when v is itself a Kronecker product):

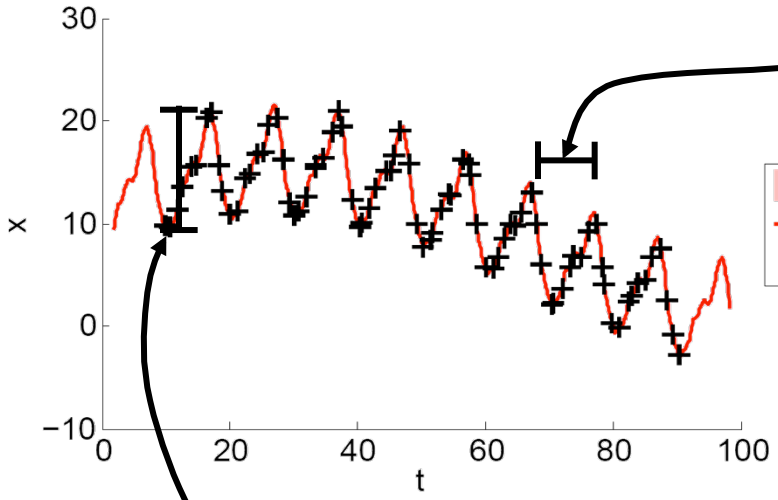
$$x = \overbrace{\left(K_a \otimes K_b\right)}^{\text{size } n_a \times n_b}{}^{-1}\left(v_a \otimes v_b\right)$$

$$= \left(K_a{}^{-1} v_a\right) \otimes \left(K_b{}^{-1} v_b\right)$$

size n_a size n_b

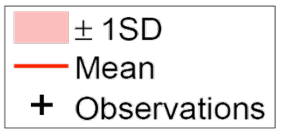
Recall that solving operations are typically $O(n^3)$!

Associated with our covariance function (and also our mean function) are a number of hyperparameters φ, such as

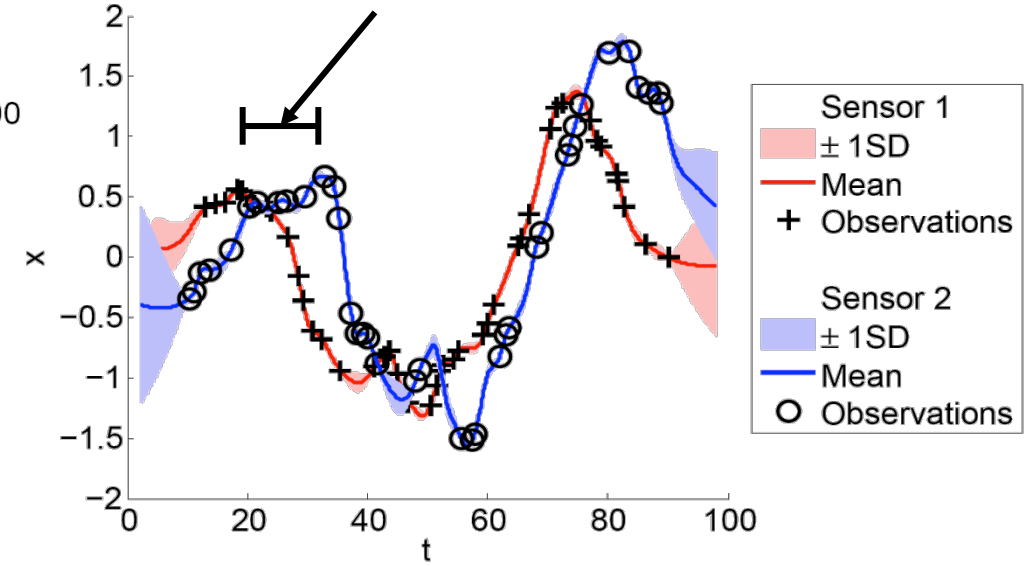


periods

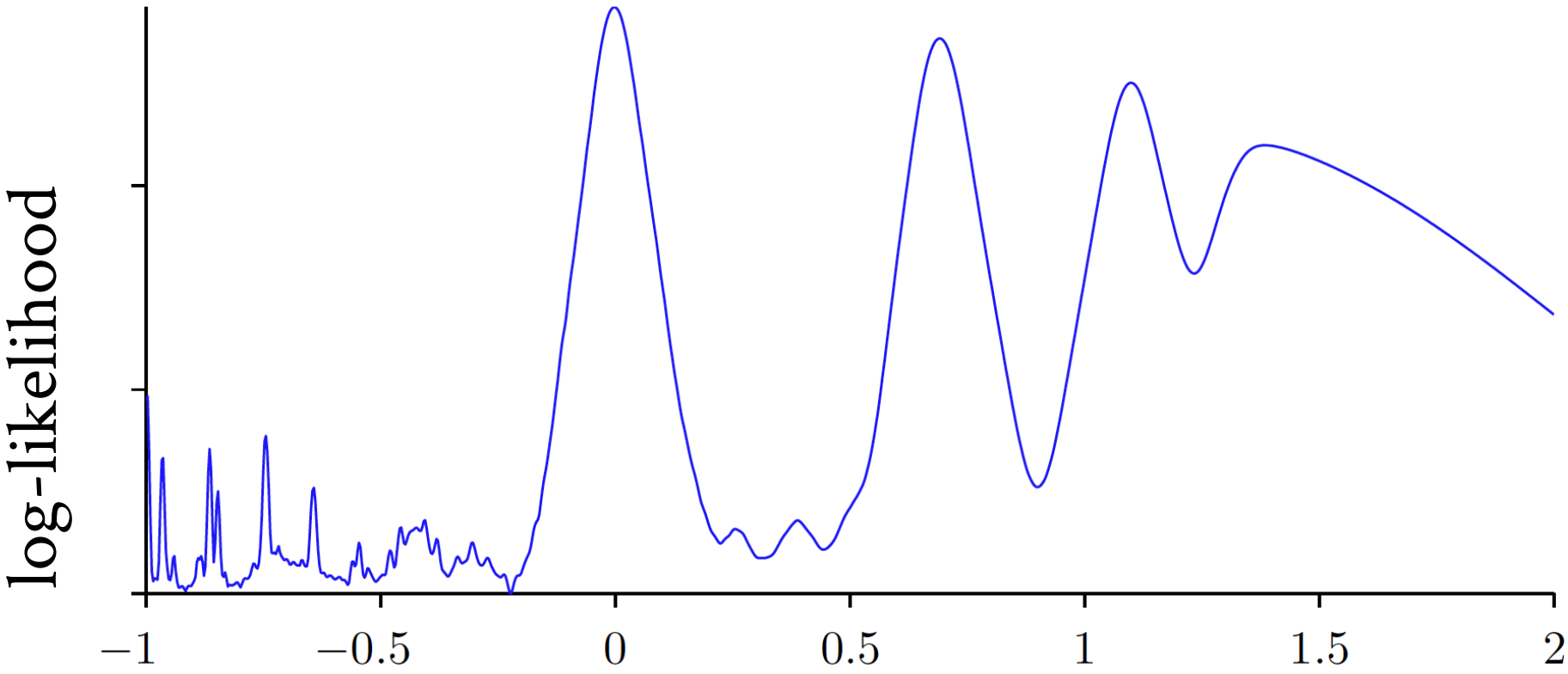
amplitudes



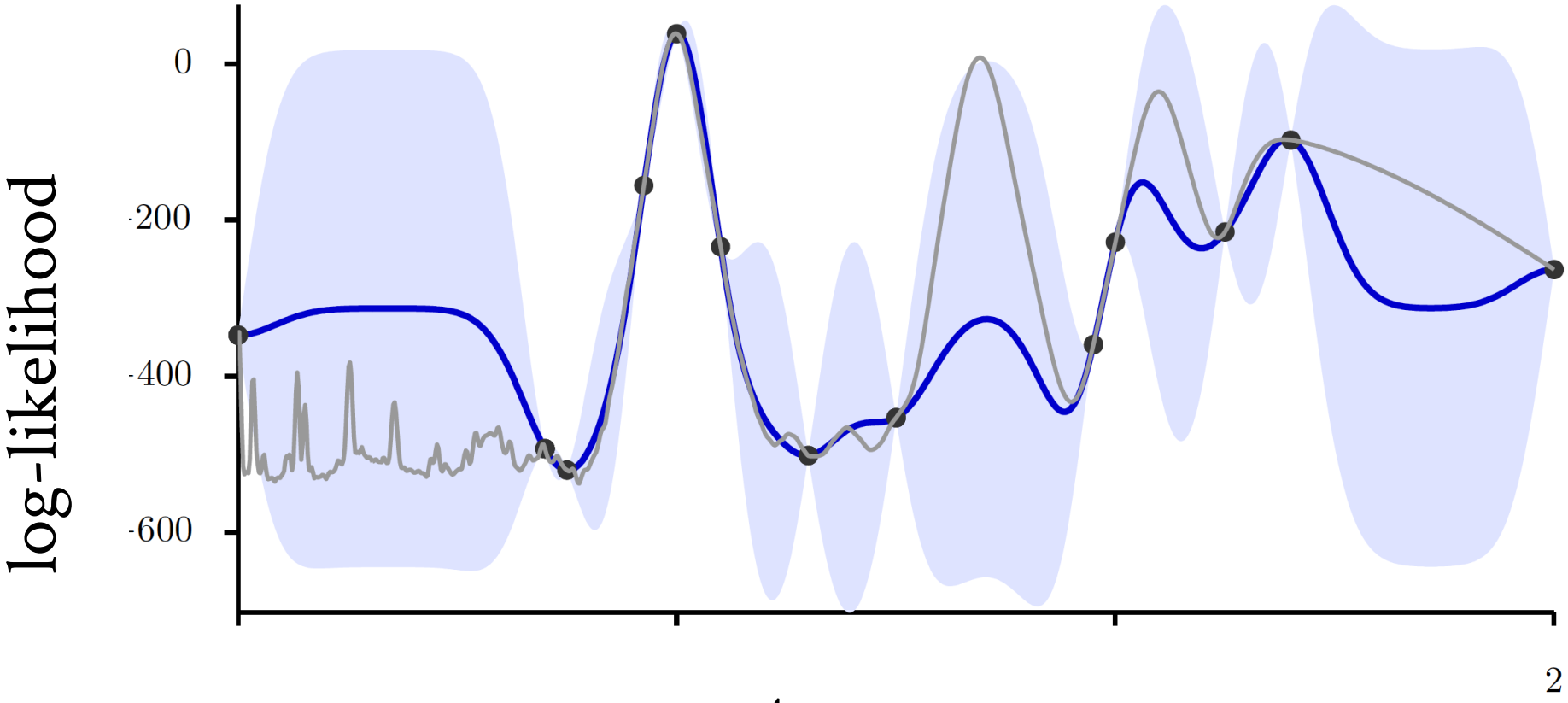
correlations



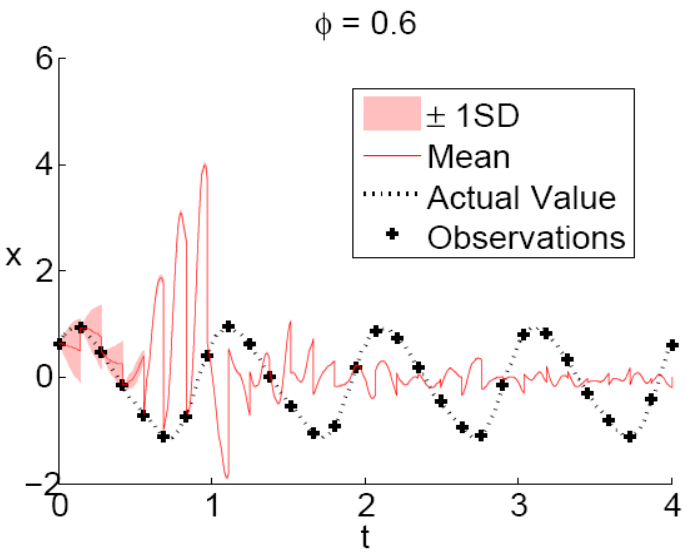
Integrating over (marginalising) these unknown hyperparameters is usually non-analytic, and requires quadrature.

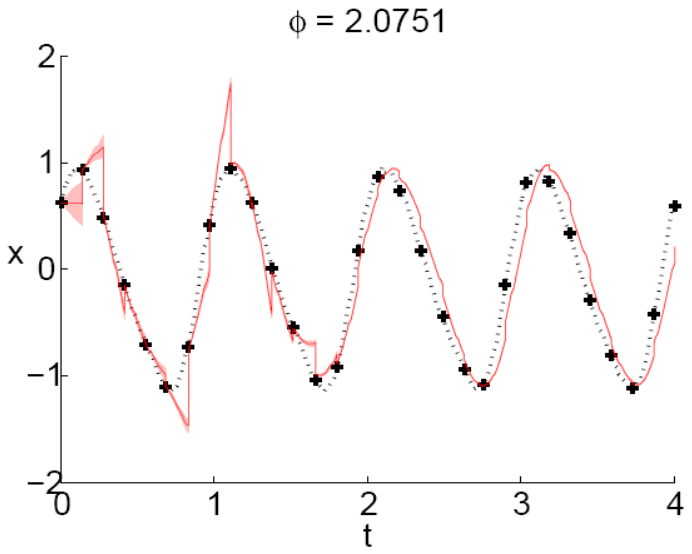


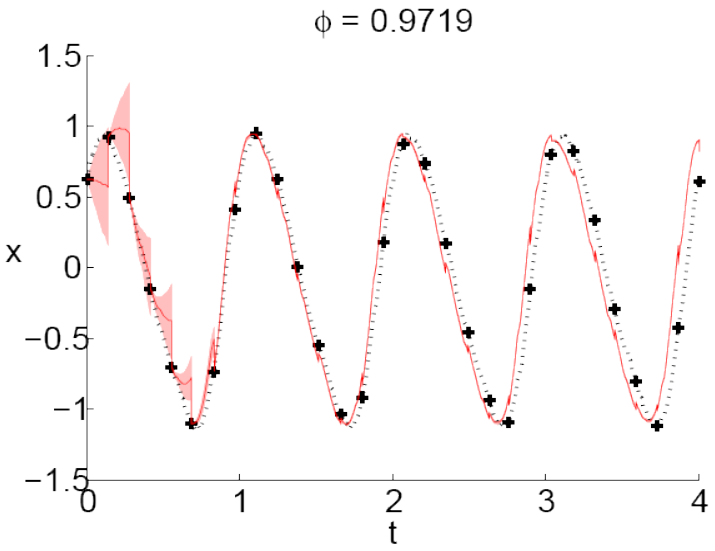
Re-cap from yesterday: Bayesian quadrature gives an excellent method for estimating the integrand – a Gaussian process.



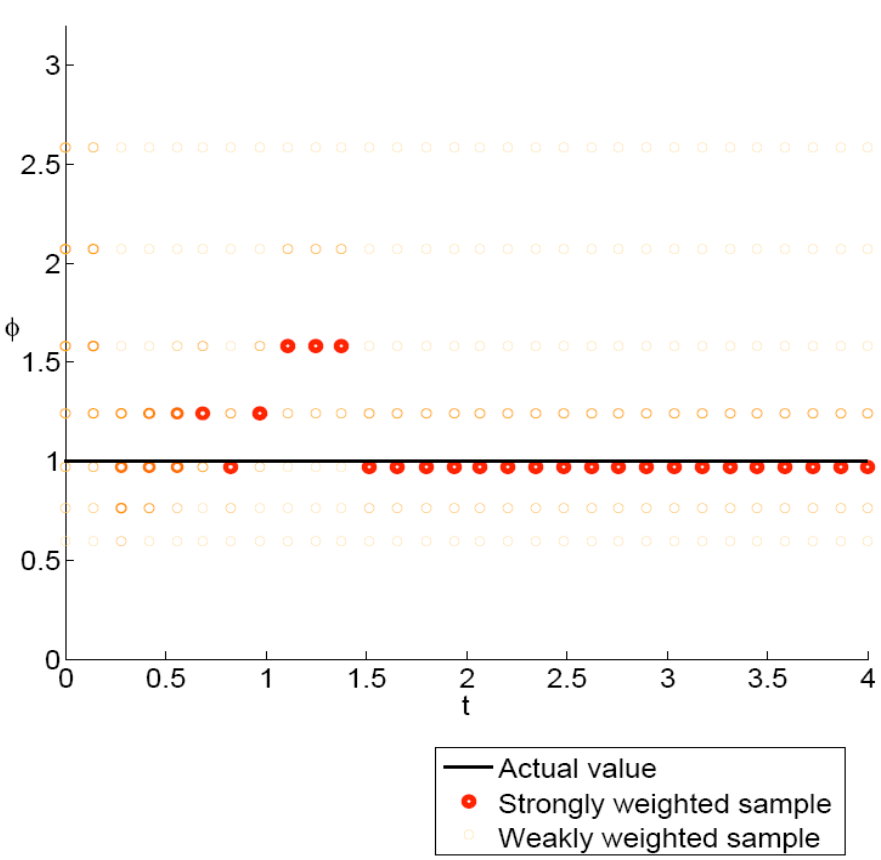
As an example, consider a fixed sample set of hyperparameters φs,
e.g. φ = the period of $x(t)$.

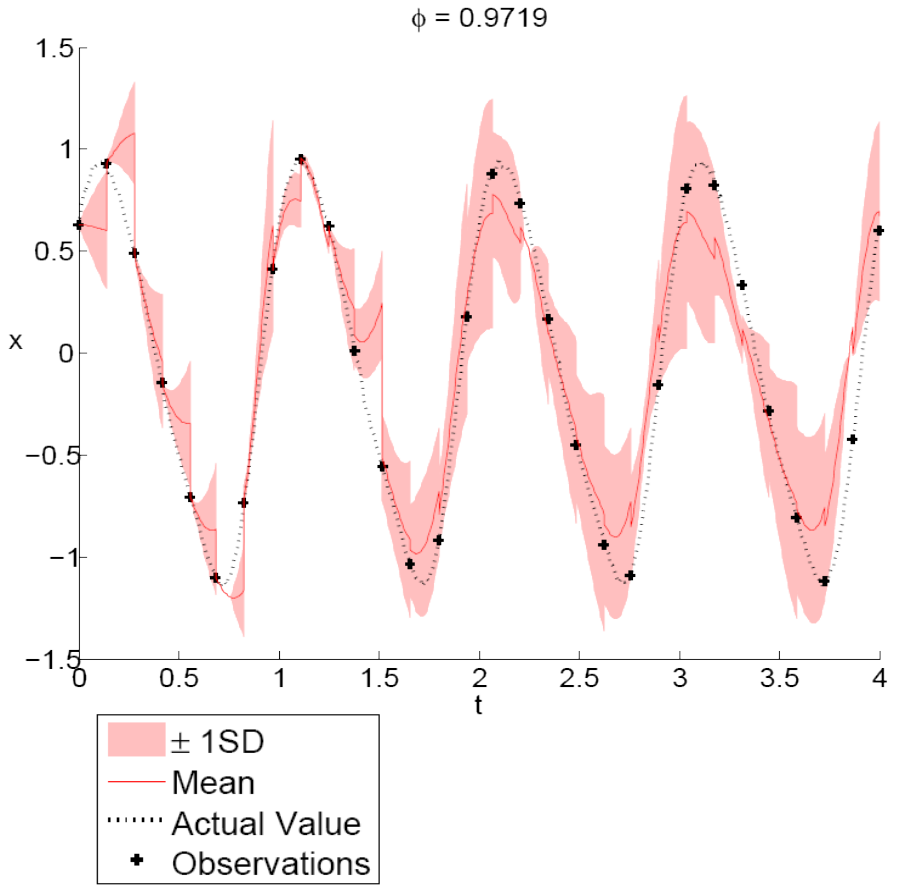




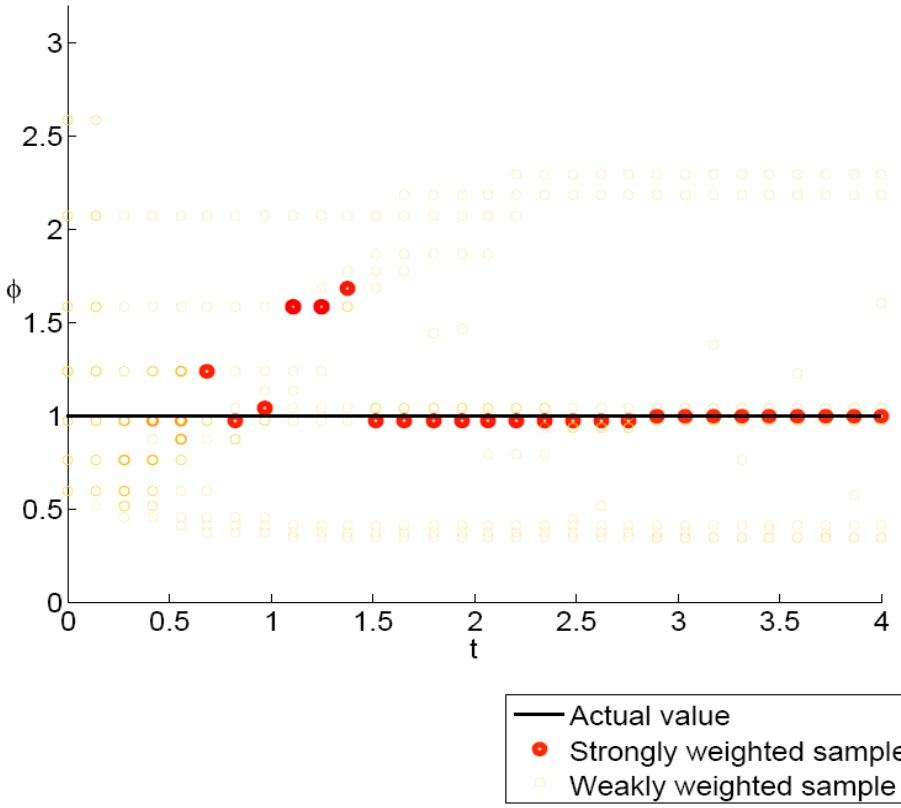


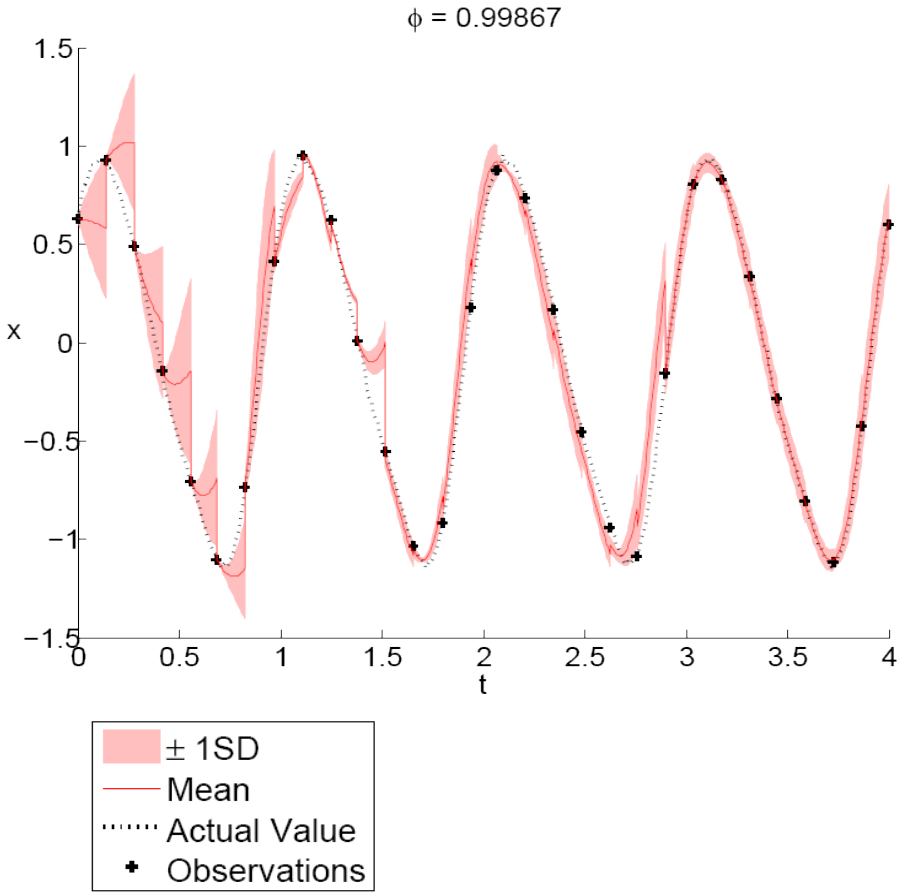
We propagate through time one Gaussian process for each of our sample set φs, adjusting the weights according to the data as we go.





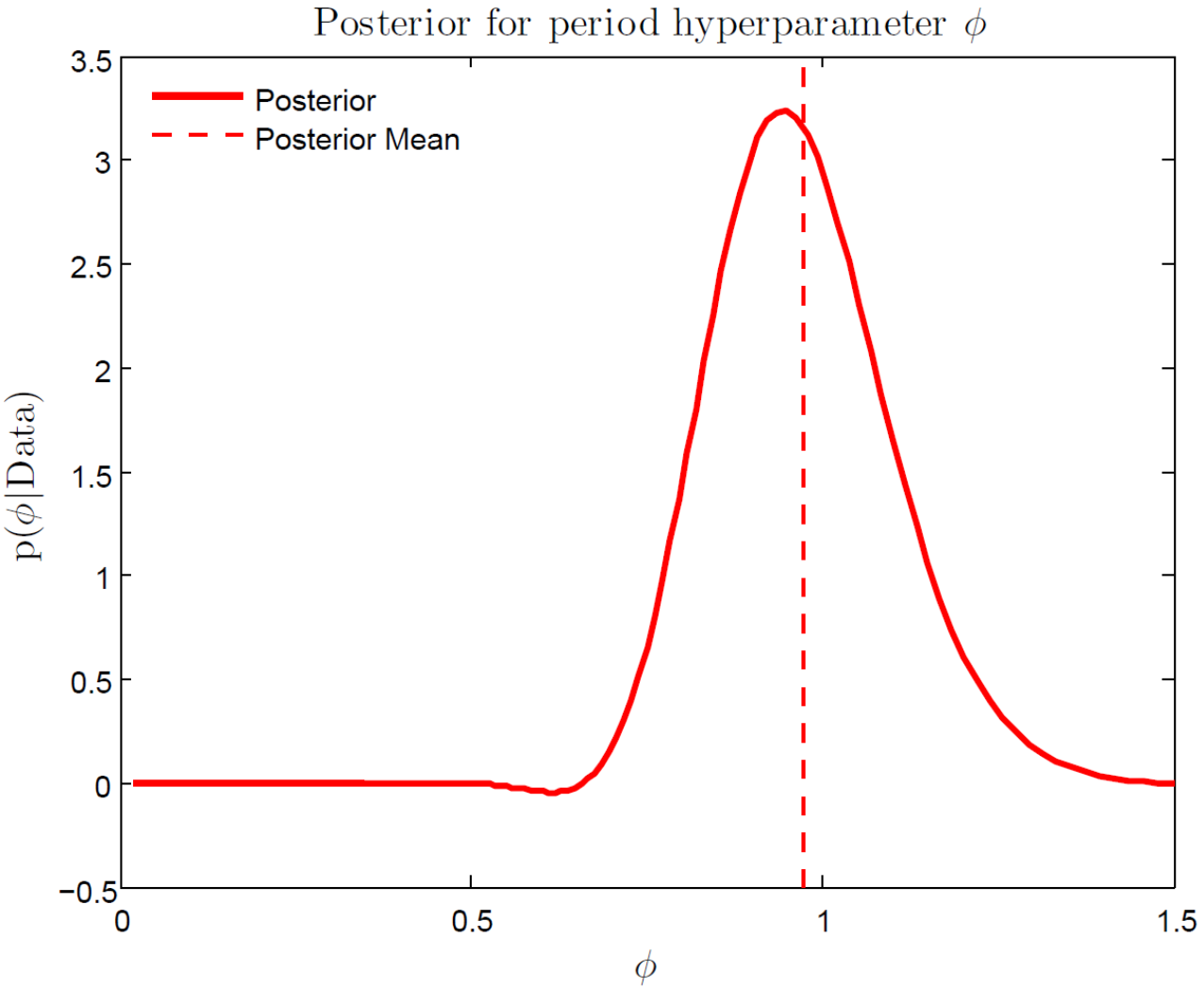
Using doubly-Bayesian quadrature, we can select samples of a period in order to perform inference for a periodic function of unknown period.



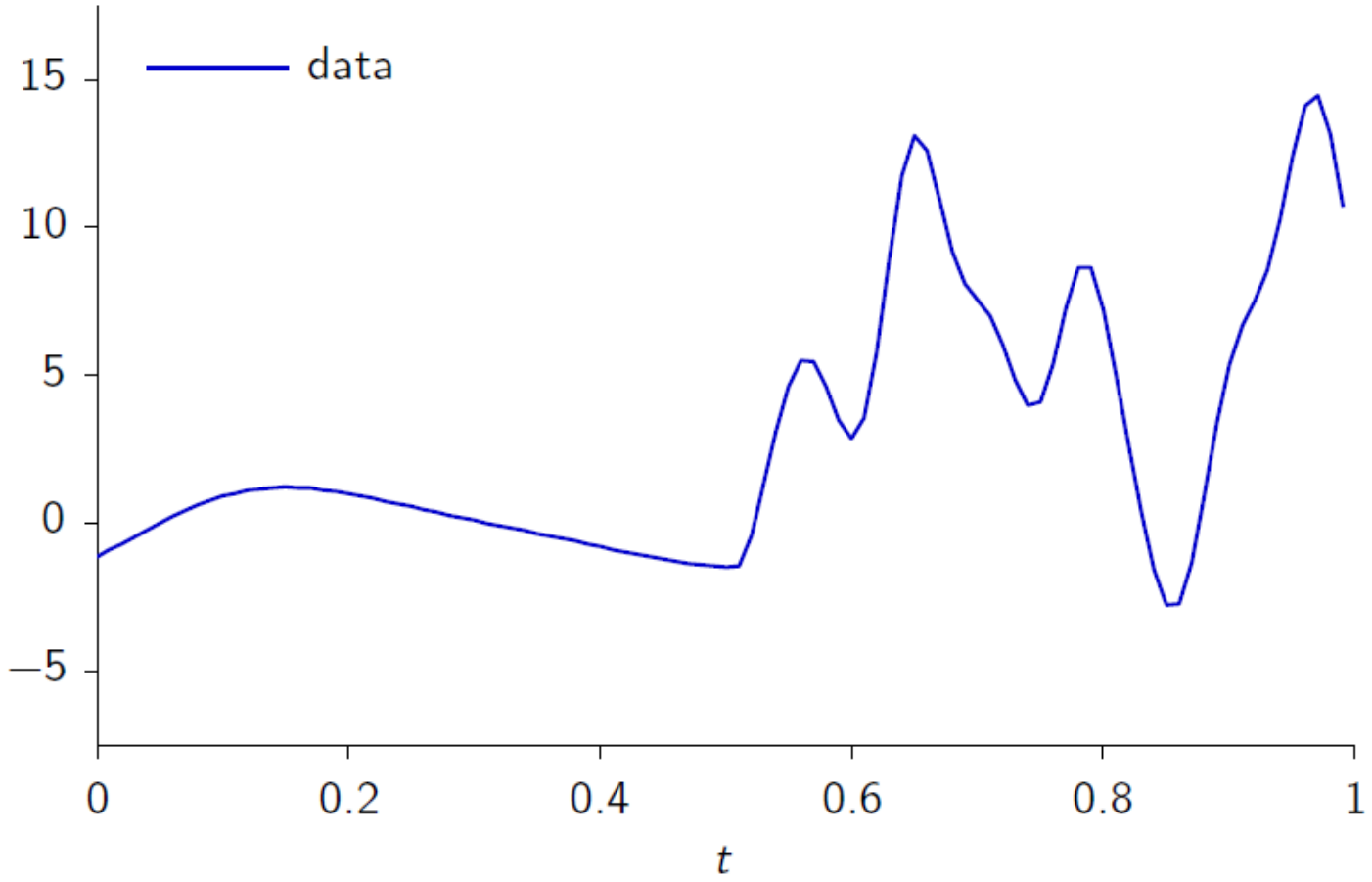


UNIVERSITY OF OXFORD

With Bayesian quadrature, we can also estimate the posterior distributions for any hyperparameters.

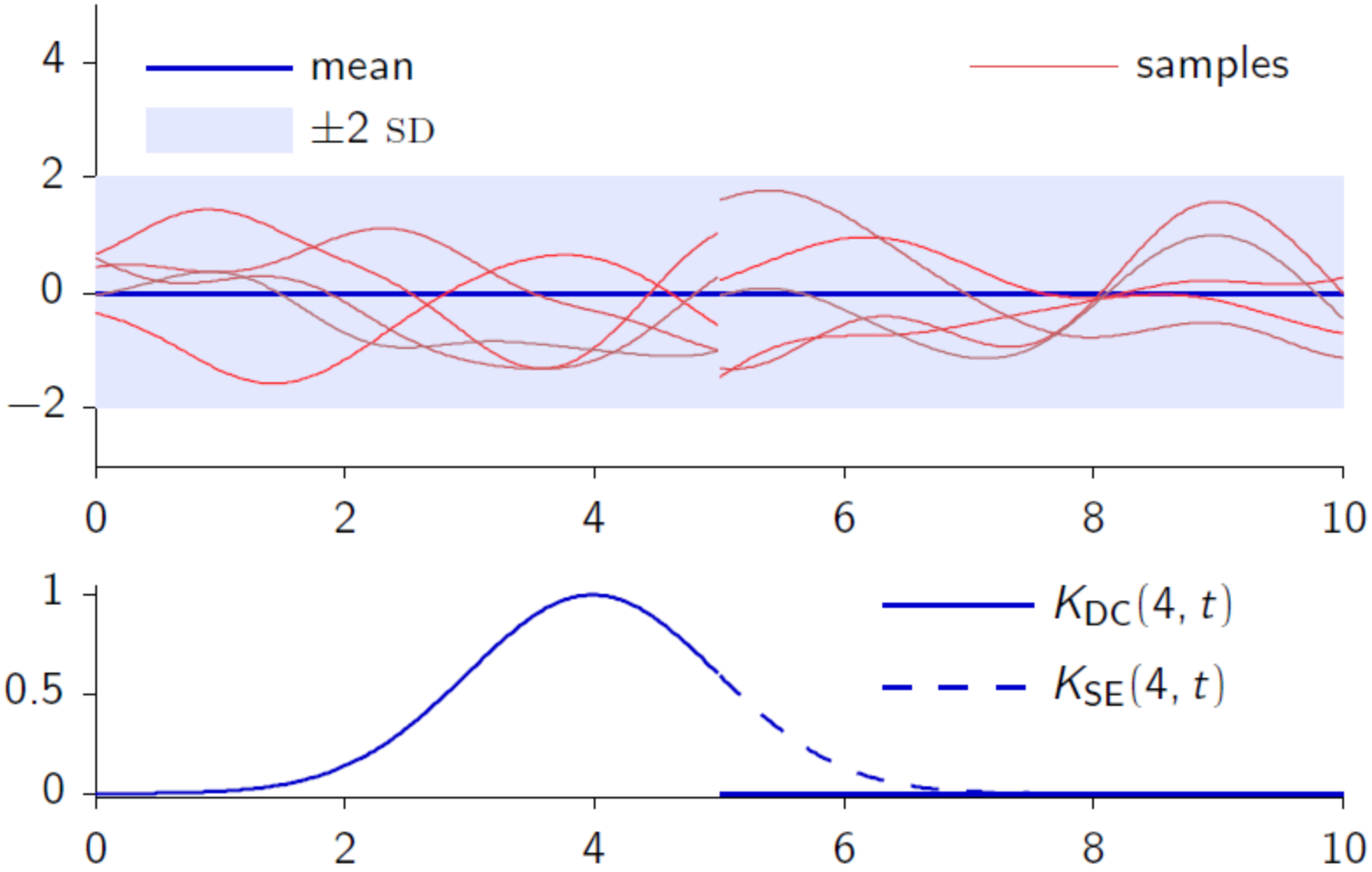


We now extend our Gaussian process framework to changepoints and faults.



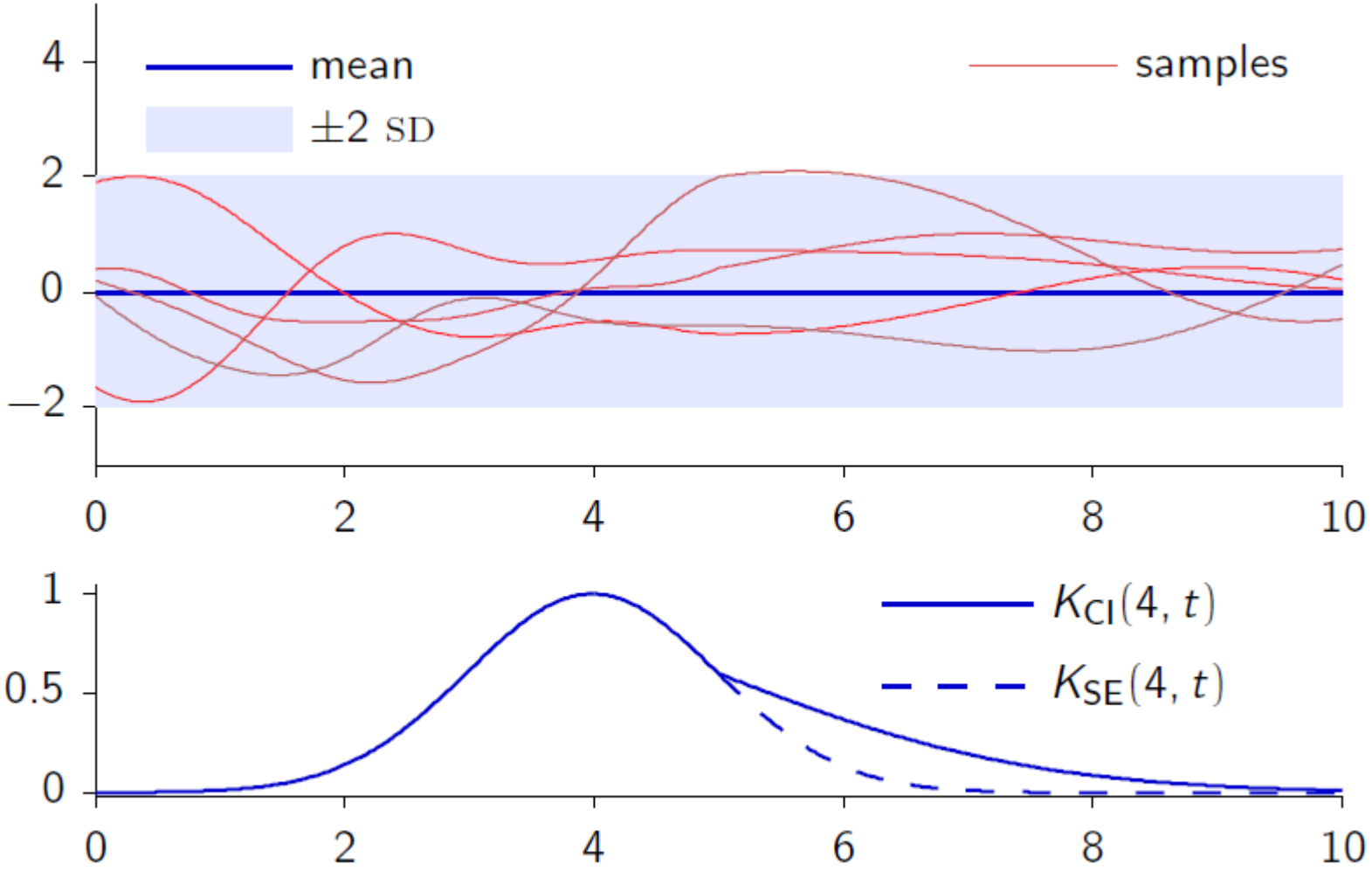
UNIVERSITY OF OXFORD

We introduce covariances for drastic changepoints.

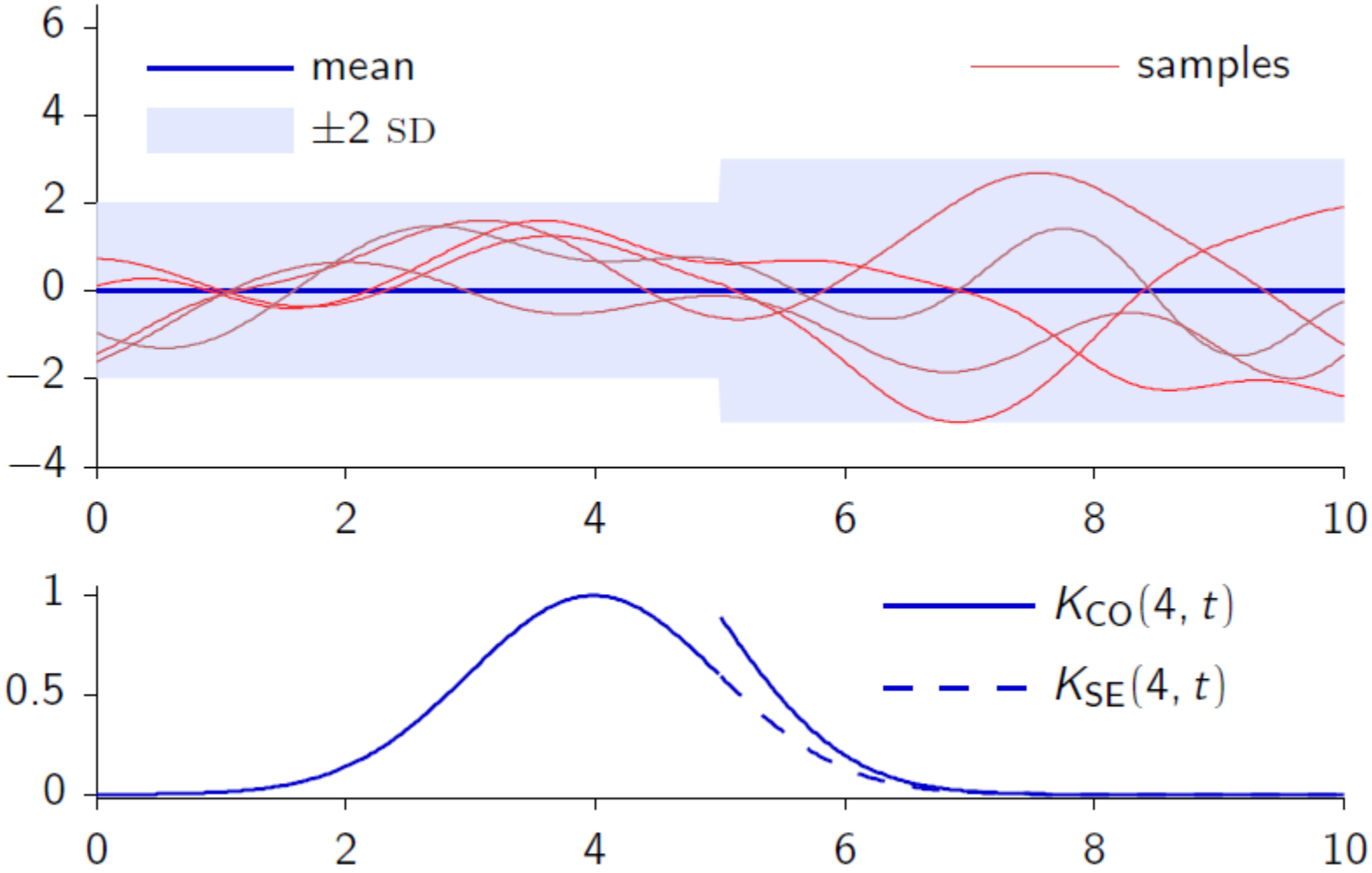


UNIVERSITY OF OXFORD

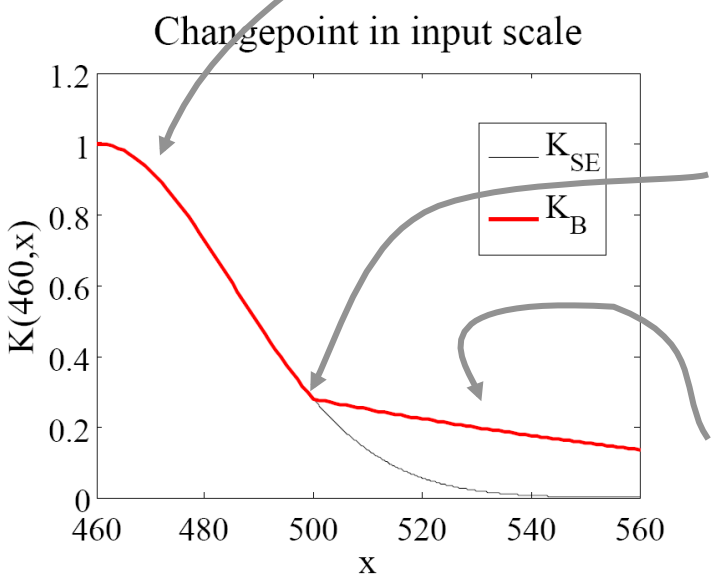
We introduce covariances for changepoints in input scale.



We introduce covariances for changepoints in output scale.



Changepoint covariances feature hyperparameters, for which we can produce posterior distributions using quadrature.



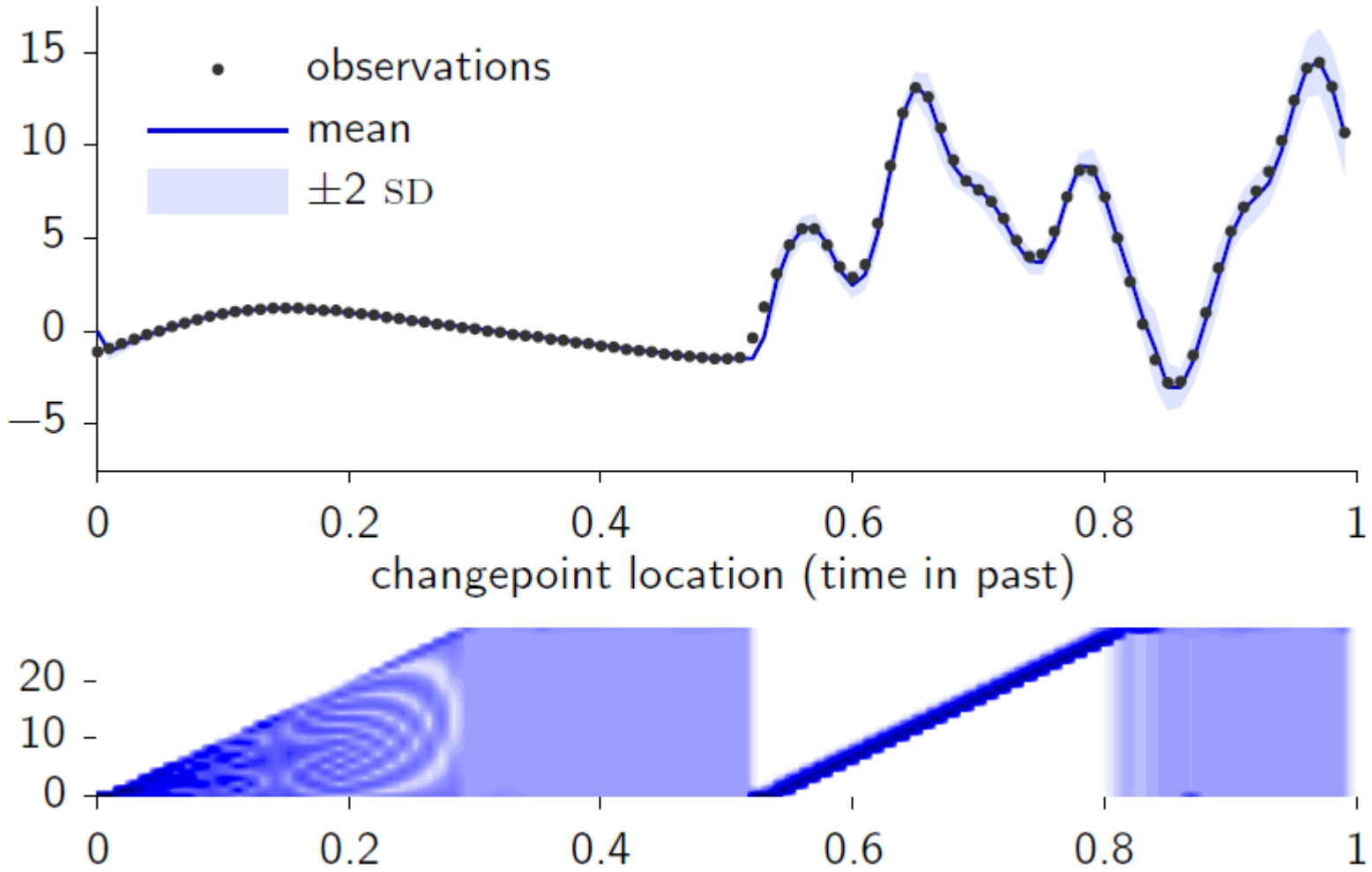
input scale pre-changepoint

changepoint location

input scale post-changepoint

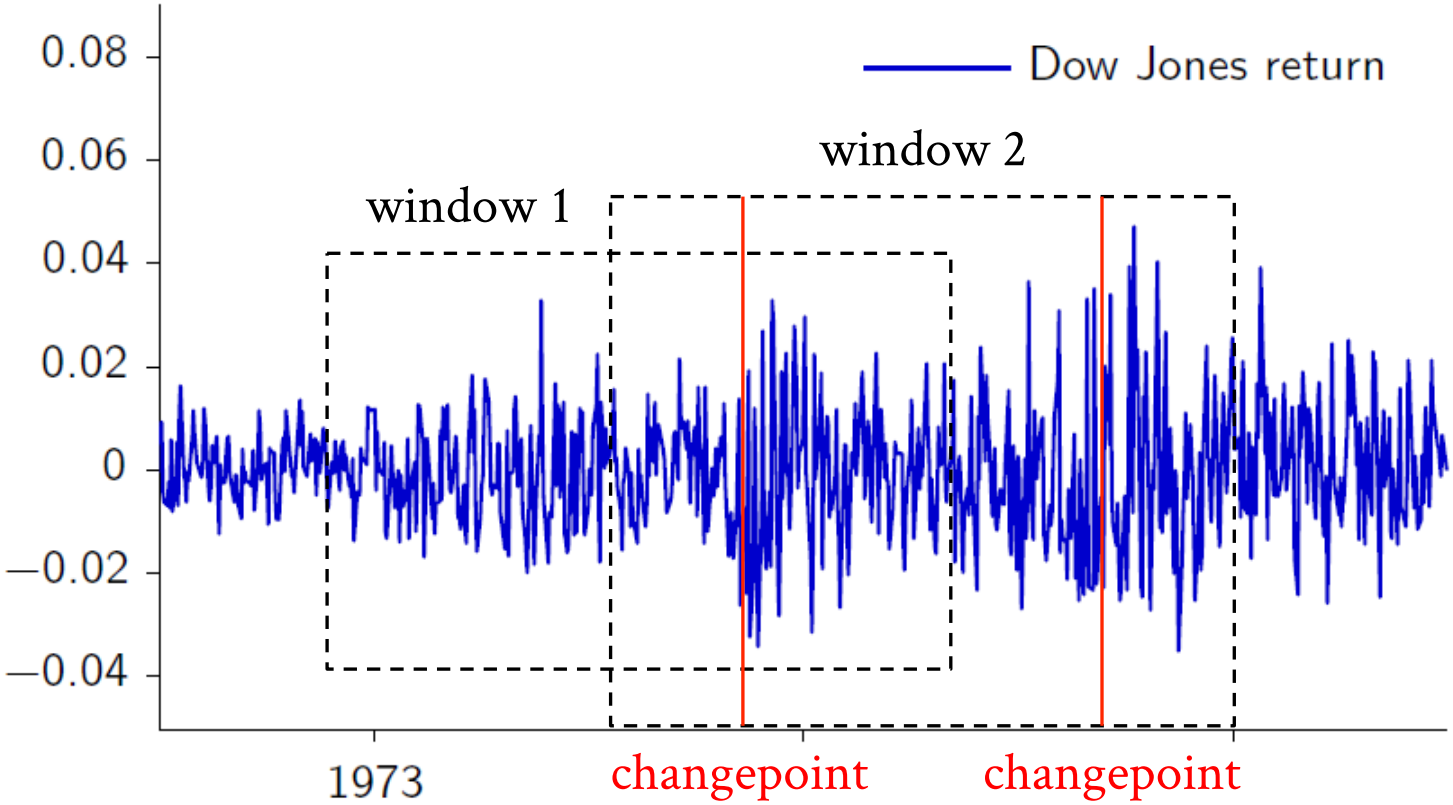
Changepoint detection requires the posterior for the changepoint location hyperparameter.

We can hence perform both prediction and changepoint detection.

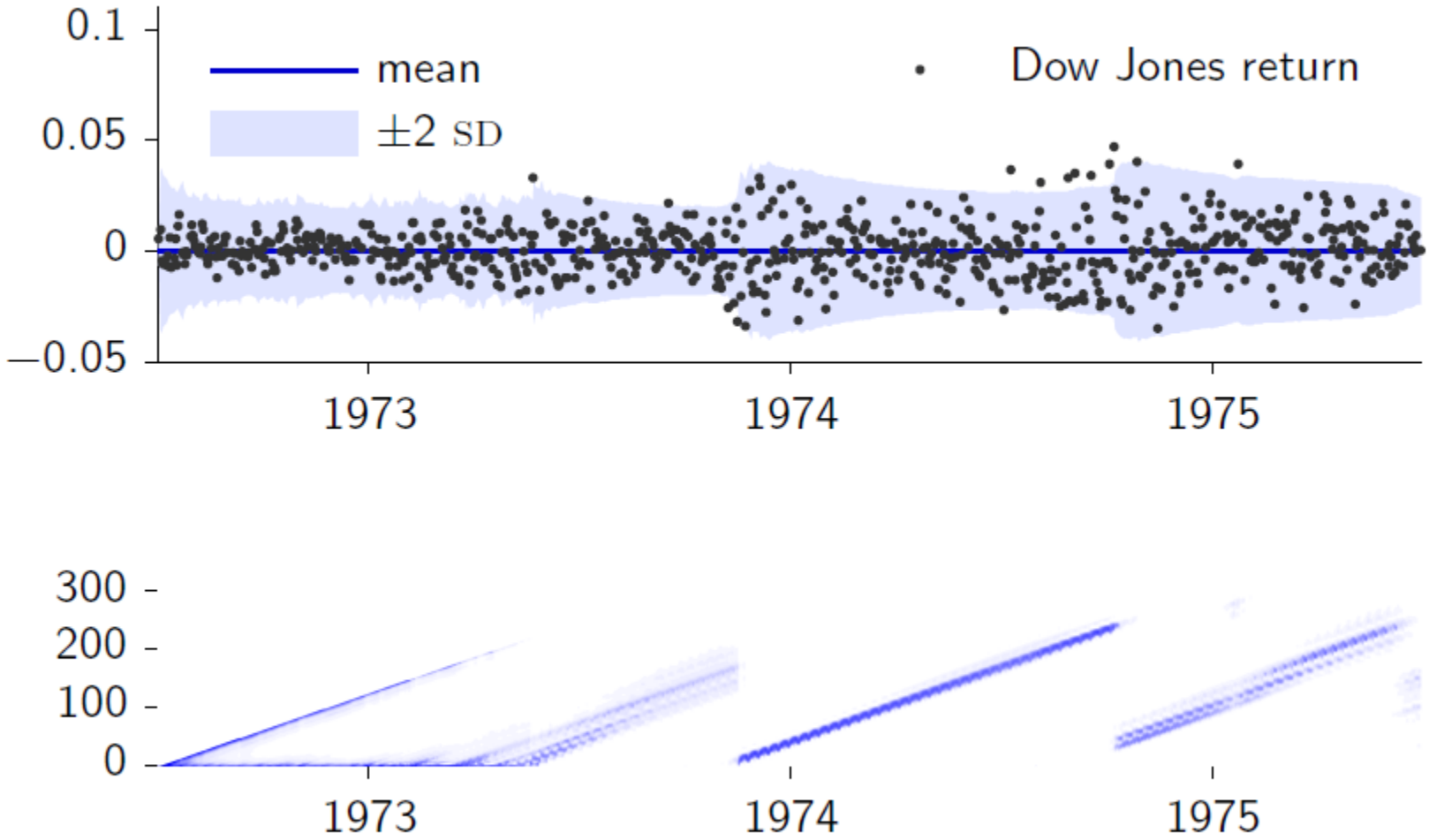


UNIVERSITY OF OXFORD

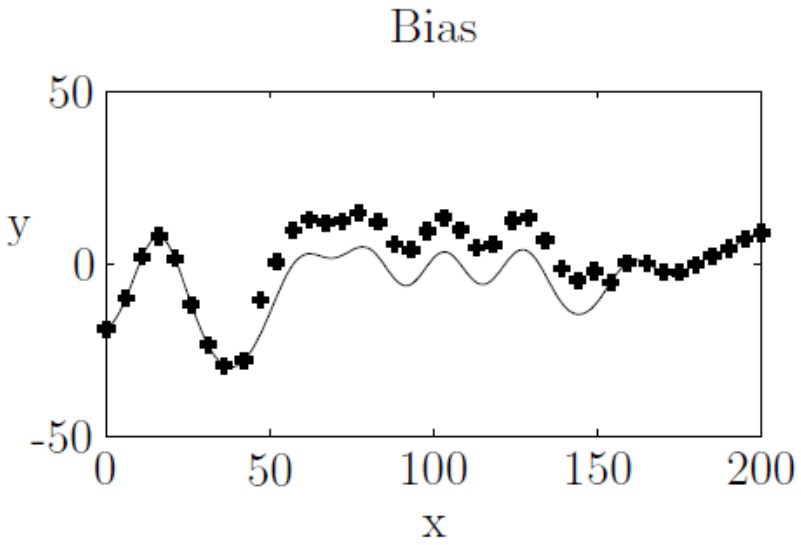
As data is usually weakly correlated across a changepoint, we usually need only consider a few changepoints in a window.

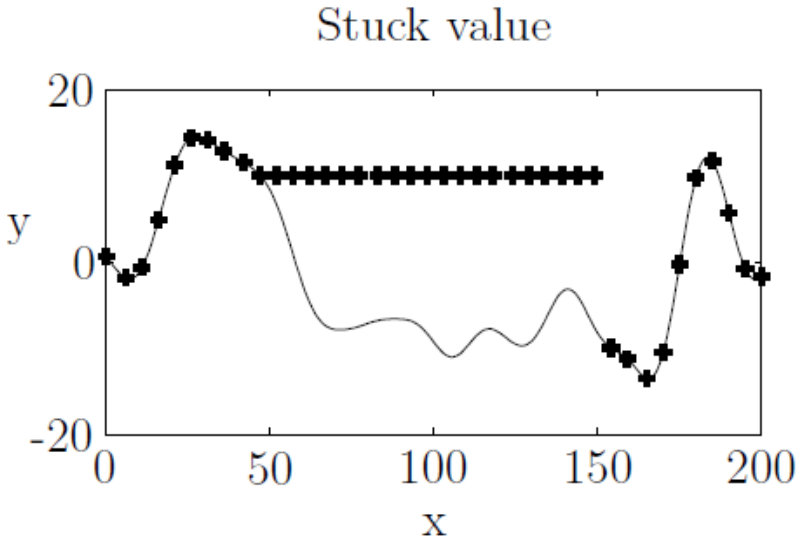


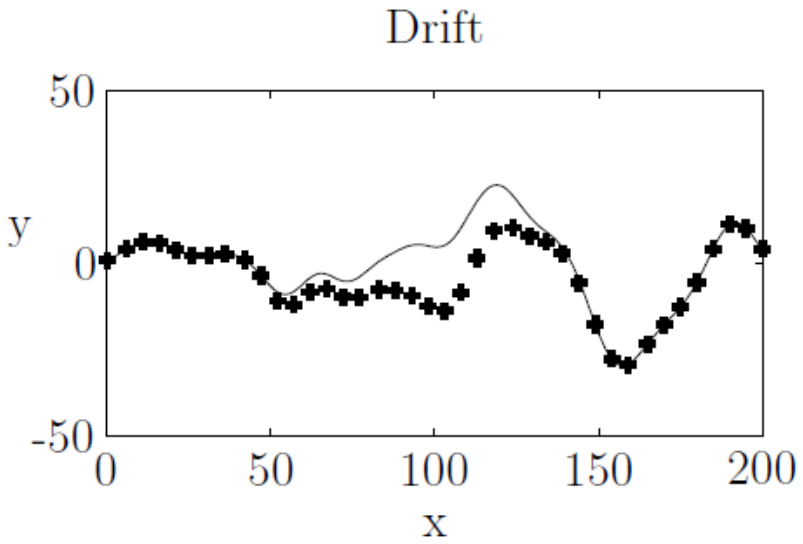
We can identify the OPEC embargo in October 1973 and the resignation of Nixon in August 1974.



A fault is defined as a change in our observation model.

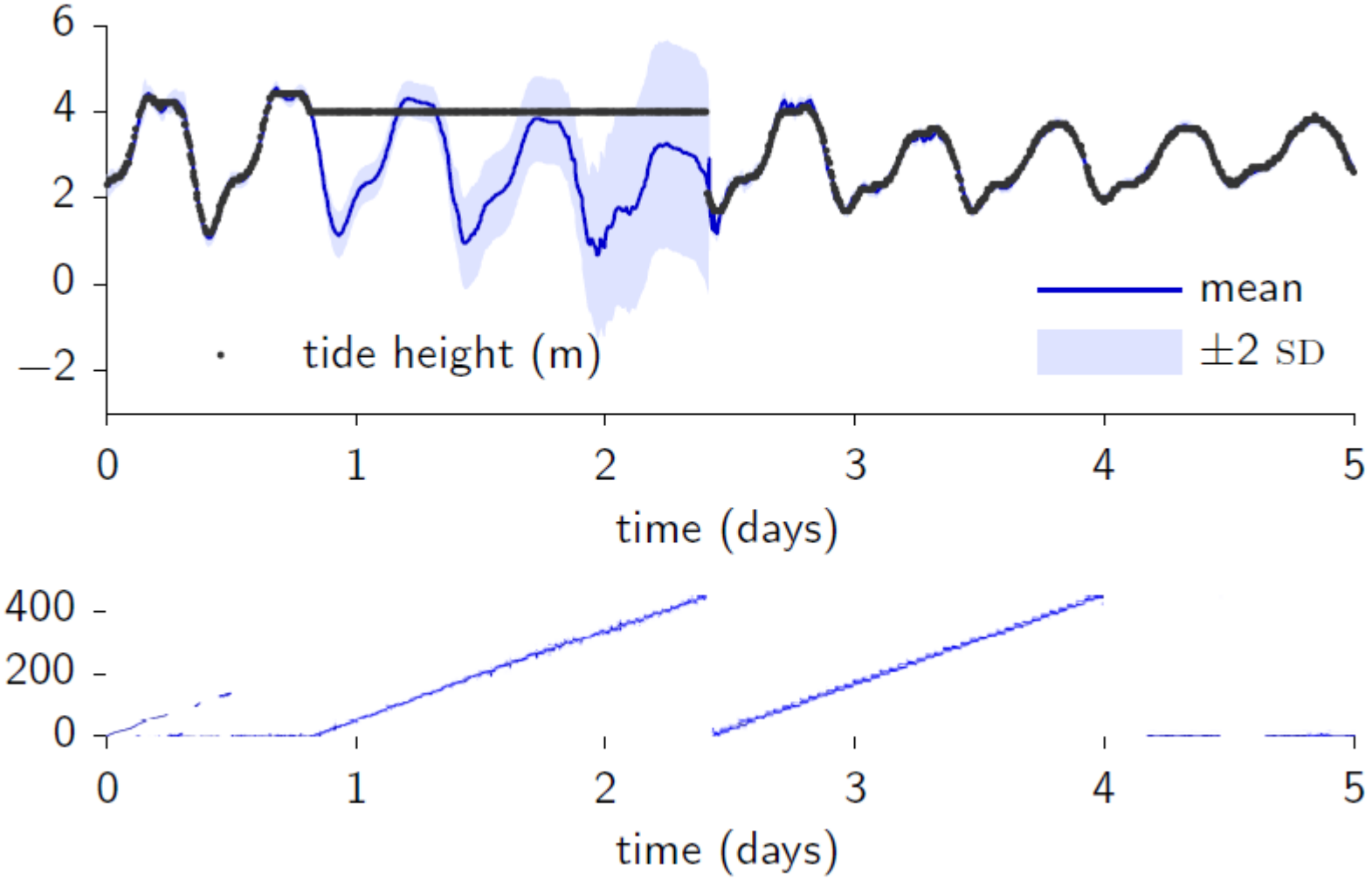




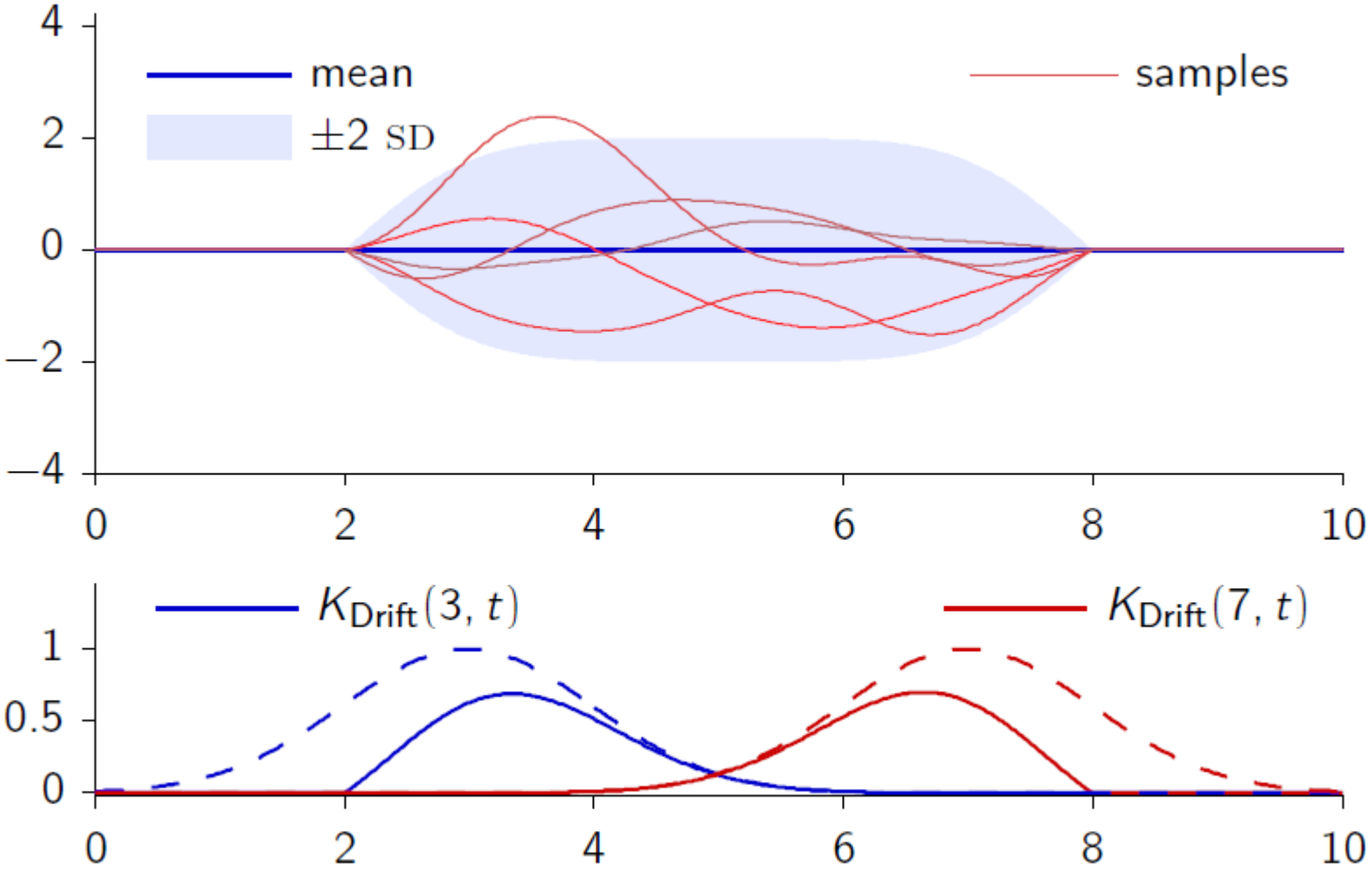


UNIVERSITY OF OXFORD

We can perform prediction even when we have uninformative faulty observations.

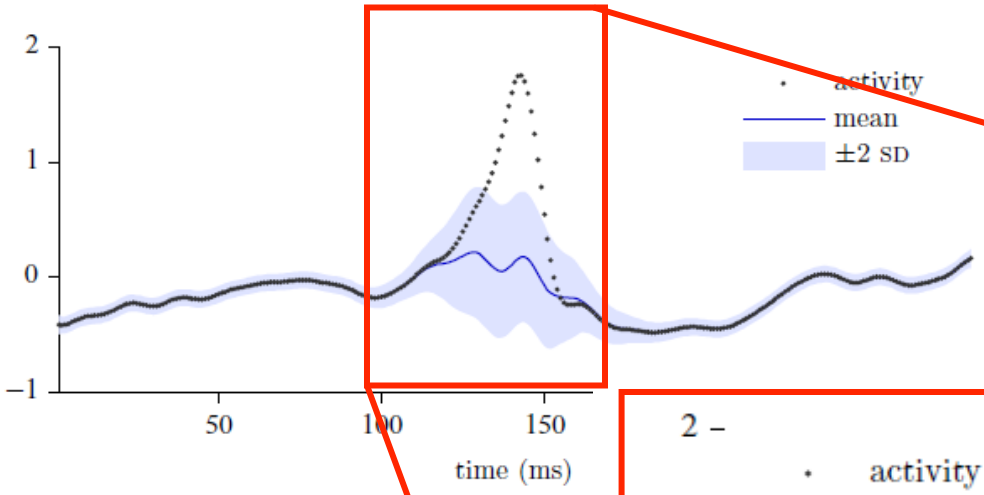


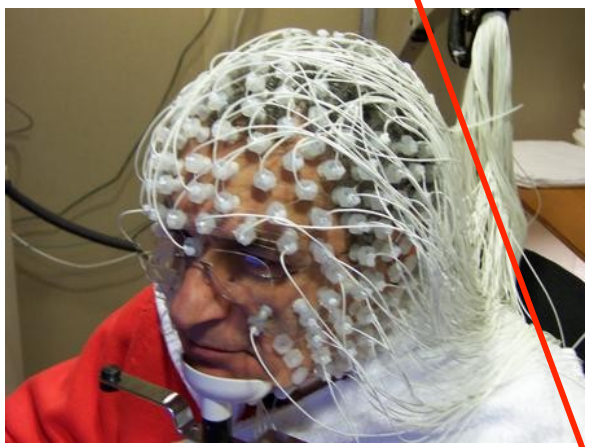
We introduce a drift covariance to model drift faults.

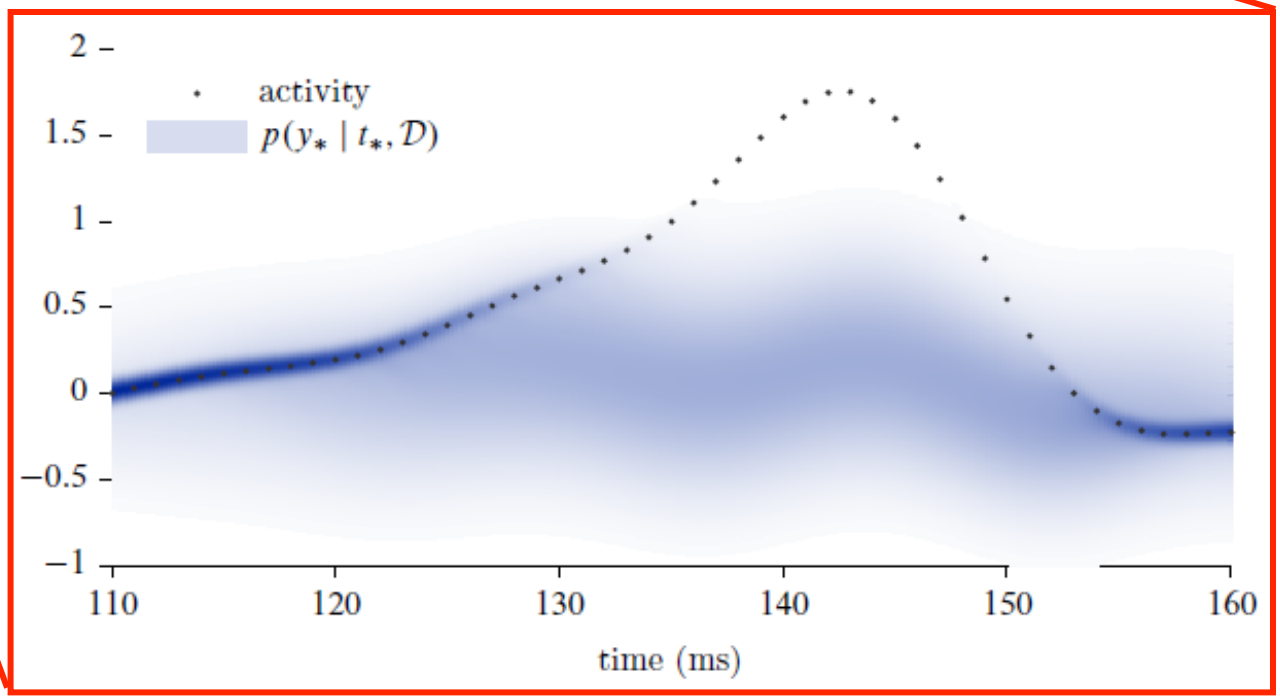


UNIVERSITY OF OXFORD

We can remove saccade faults from EEG data.

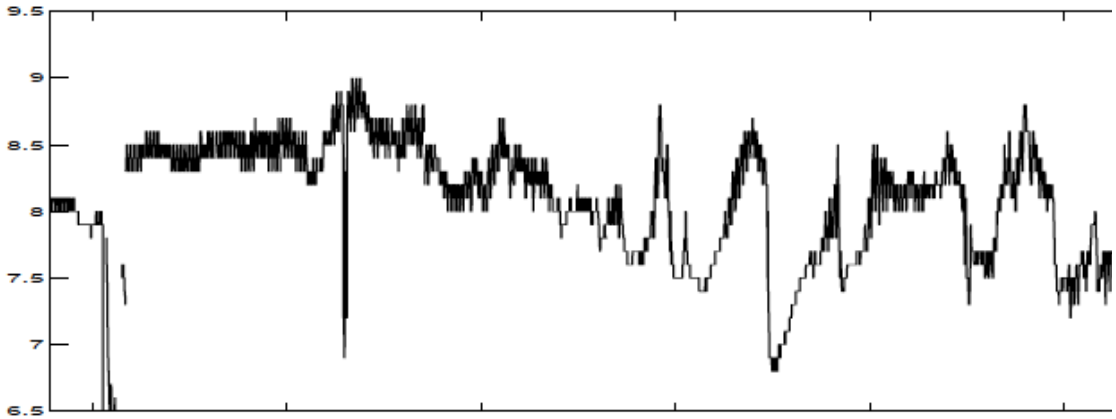






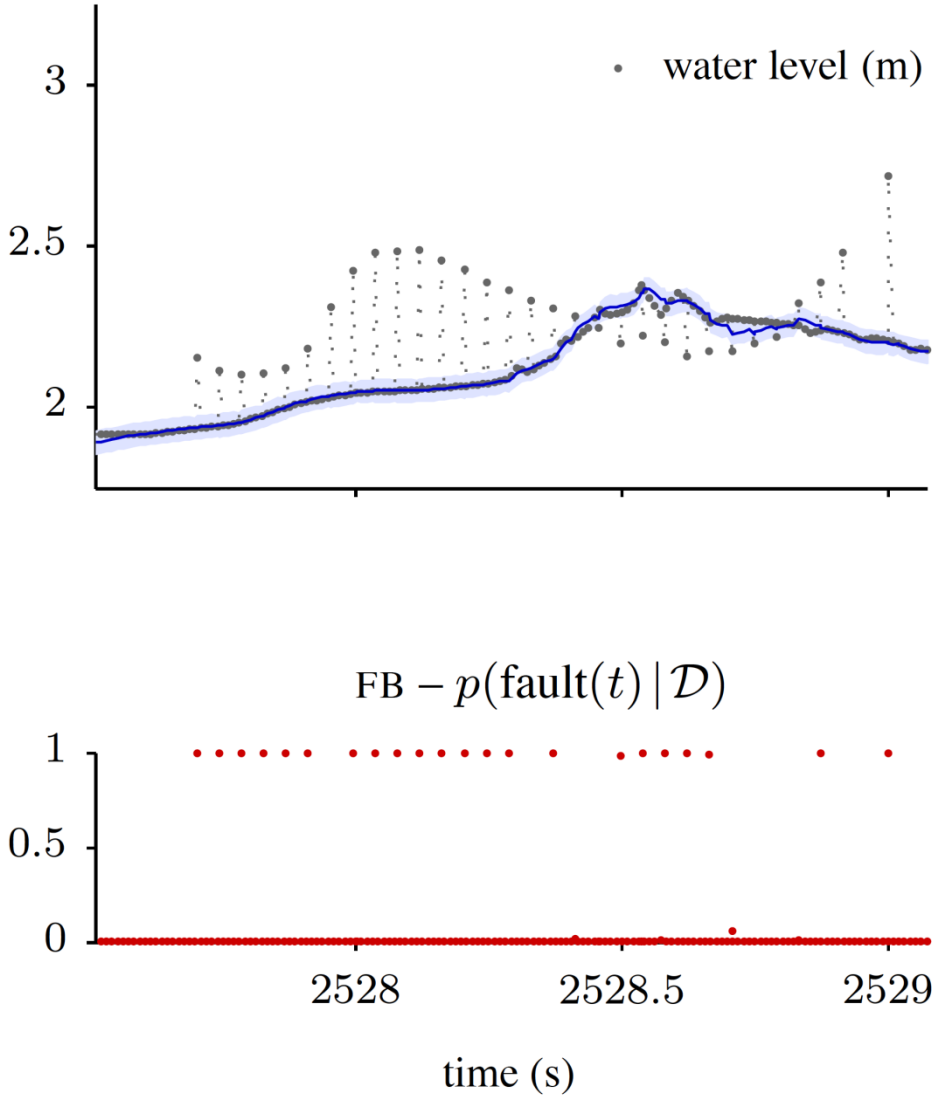
UNIVERSITY OF OXFORD

Water sustainability requires fast and effective monitoring of problematically-corrupted water data: we need a method capable of managing uncharacterised faults.



UNIVERSITY OF OXFORD

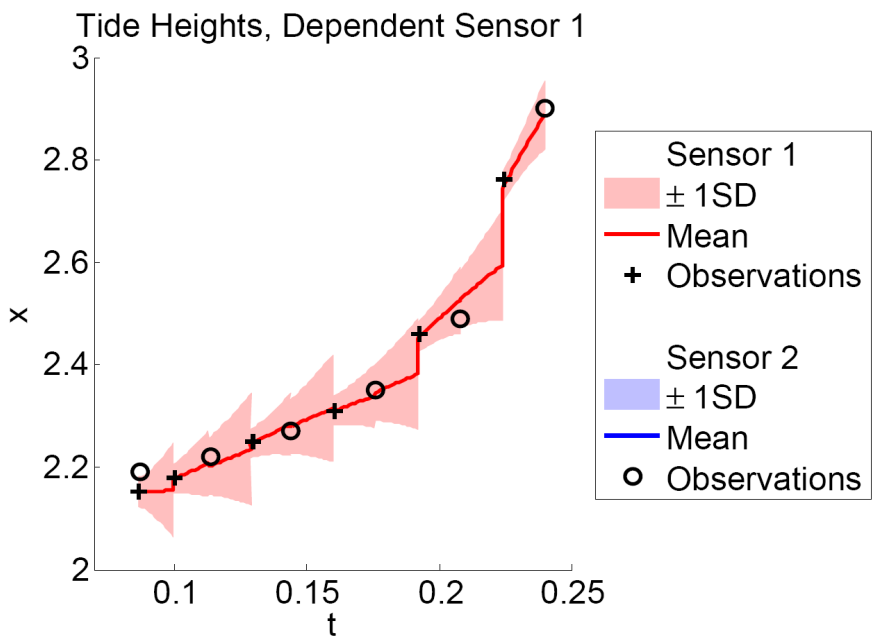
By averaging over the faultiness of each datum, we can perform effective fault removal for this data.

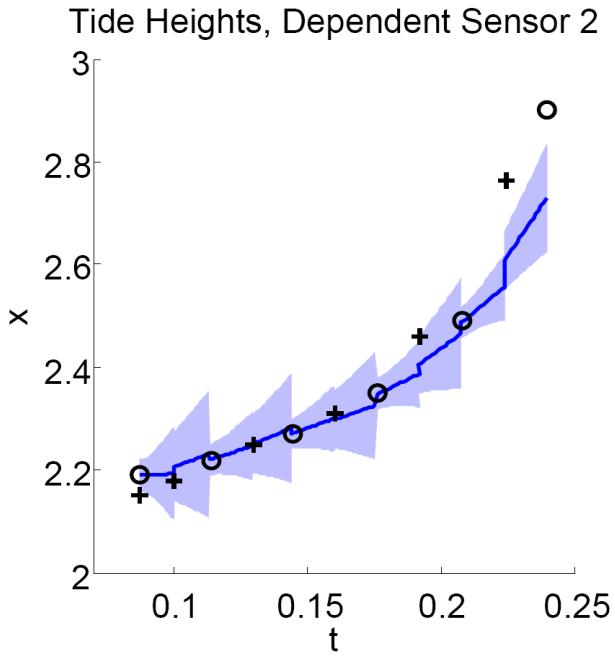


UNIVERSITY OF OXFORD

Often we wish to take only the most informative data; by defining the costs of observation and uncertainty, we can perform optimal active data selection.

We can use our predictive error bars to perform active data selection. For example, we might take a loss function that is infinite if the predictive variance for any time-series (sensor) ever exceeds some threshold.



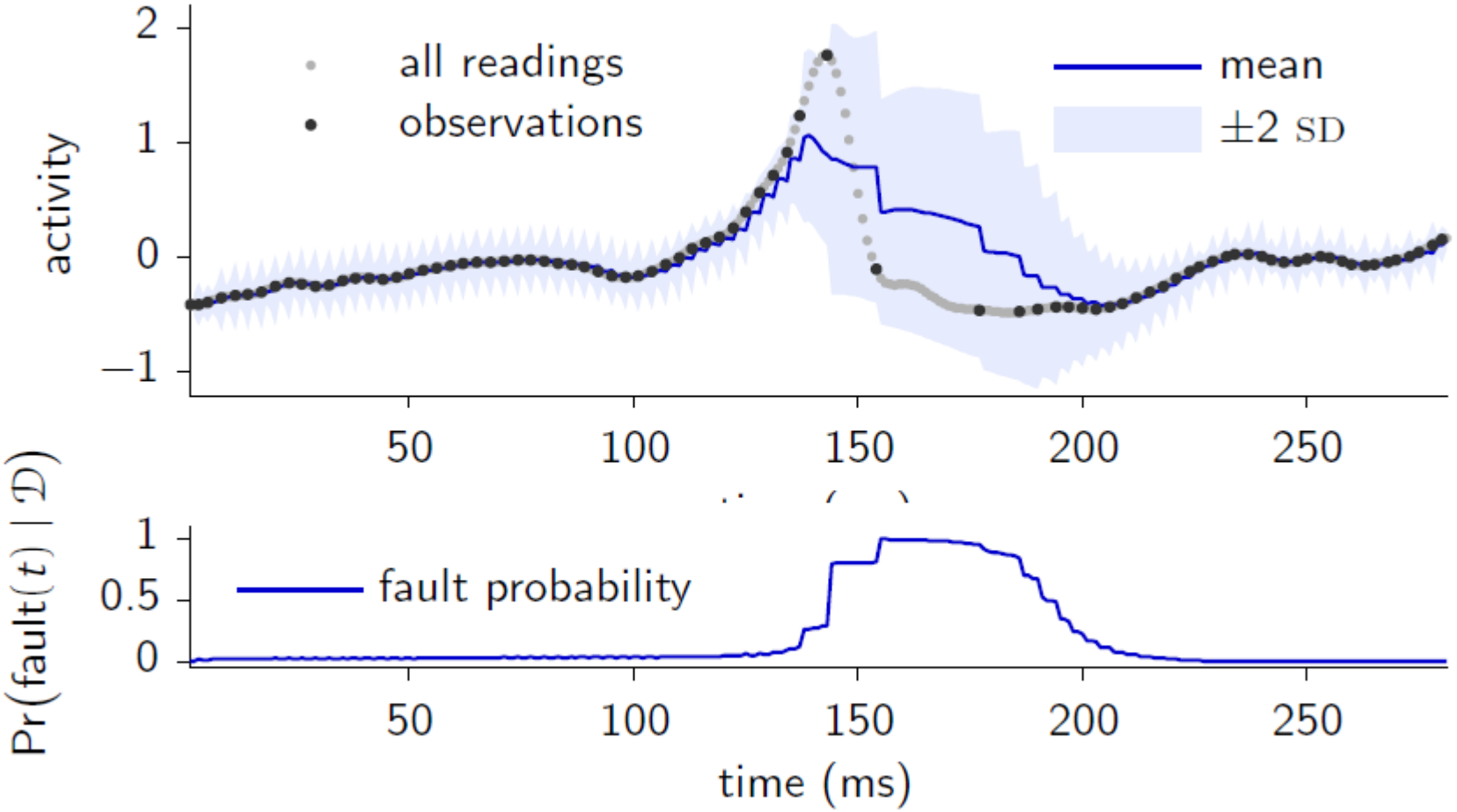


A better alternative is to take a loss function that is the sum of the predictive variances at each of our time-series (sensors), along with a cost of observation.

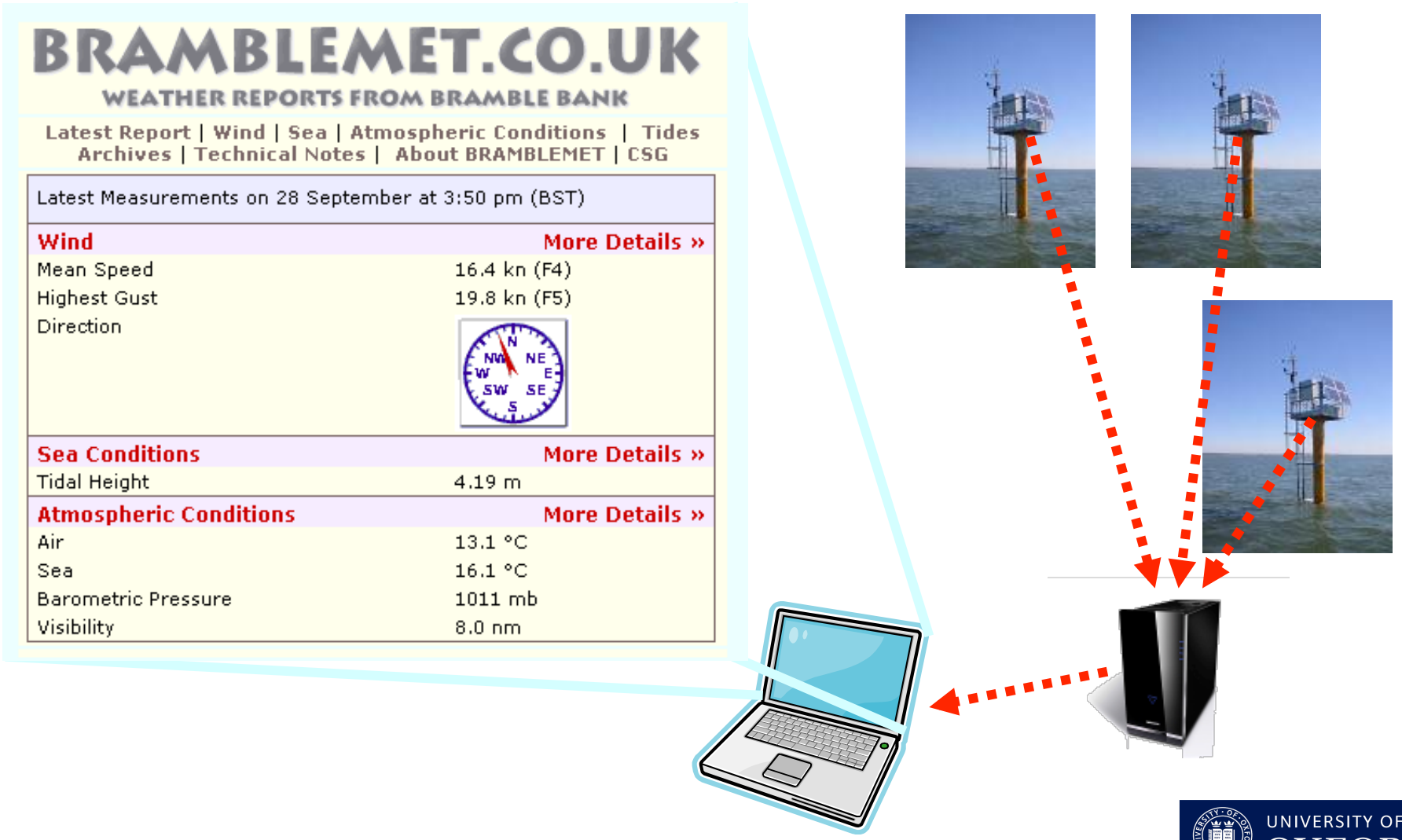
We'll then take an observation only when the resulting expected reduction in summed-variance exceeds the cost of observation.

The observation we choose will be the one that gives the greatest expected reduction in summed-variance.

Active data selection hence reduces the rate of sampling during faulty, uninformative states.



Bramblemet is a network of wireless weather sensors deployed in the Solent.



UNIVERSITY OF OXFORD

Bramblemet is used by port authorities and recreational sailors.

ASSOCIATED BRITISH PORTS

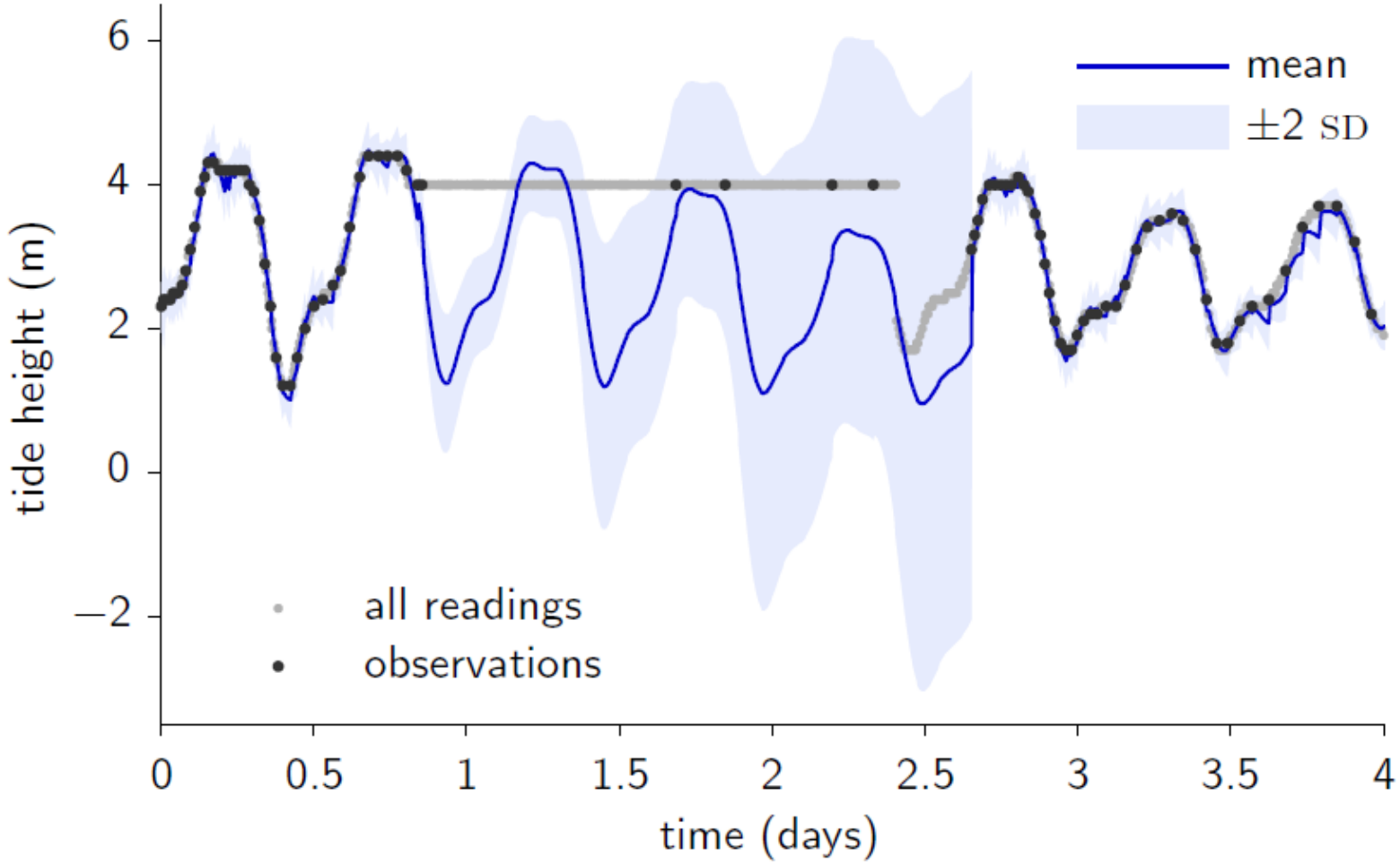
Lifeboats

Like many sensor networks, the remoteness of its sensors mean that active data selection is important.

UNIVERSITY OF OXFORD

Please enjoy the demo!

Active data selection takes fewer samples when the data is faulty.

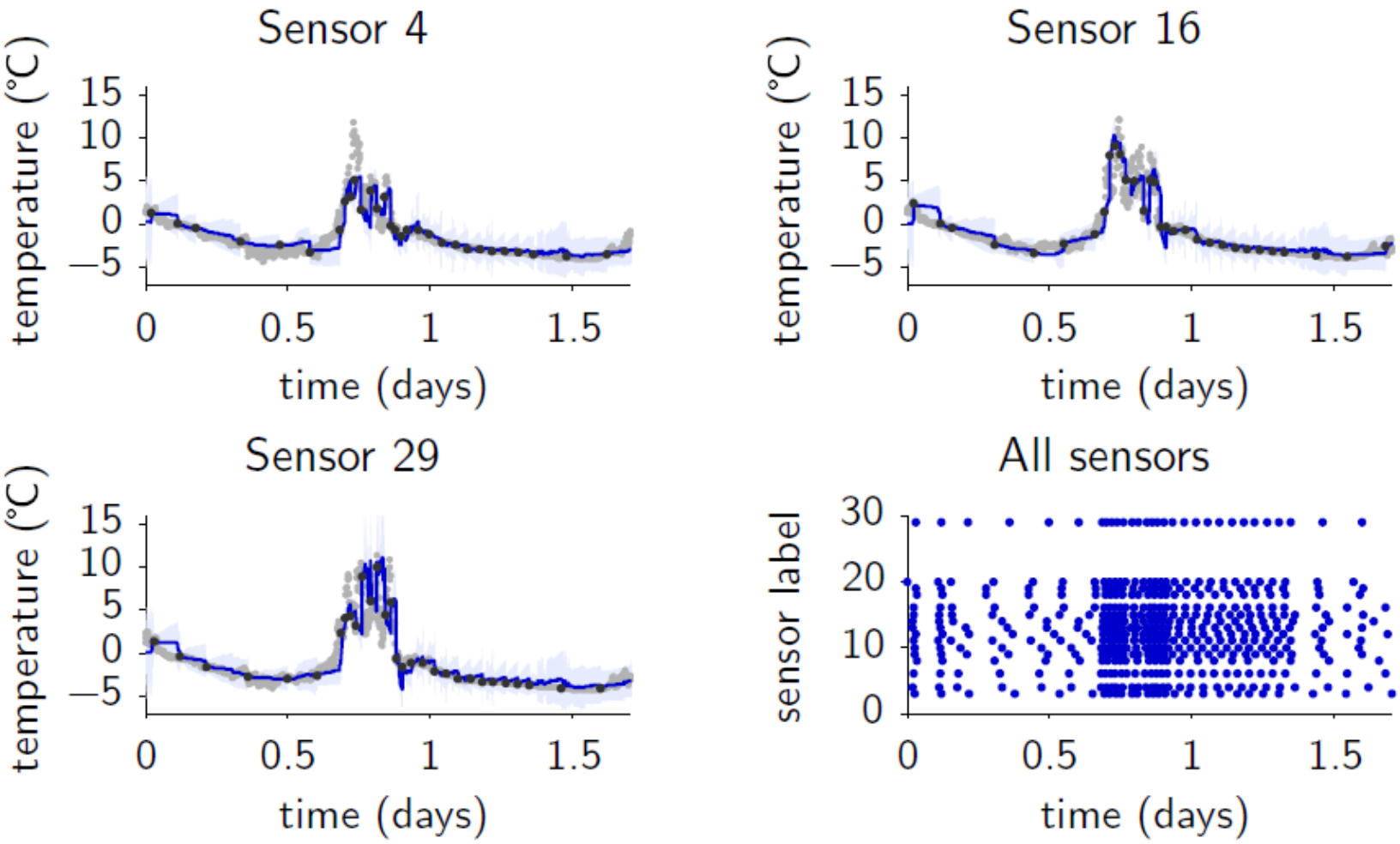


UNIVERSITY OF OXFORD

Wannengrat hosts a remote weather sensor network used for climate change science, for which observations are costly.

UNIVERSITY OF OXFORD

Our algorithm selects more observations during interesting volatile periods.



UNIVERSITY OF OXFORD

Thanks! I would also like to thank my collaborators,
Roman Garnett,
Steve Reece
Stephen Roberts
and Alex Rogers.

References

http://www.robots.ox.ac.uk/~mosb

UNIVERSITY OF OXFORD