Lecture 6: State-Space Inference in Gaussian Process Regression

GP Regression via Kalman Filtering and RTS Smoothing

Simo Särkkä

Aalto University, Finland (visiting at Oxford University, UK)

November 28, 2013
1 Gaussian process regression
2 State space representation of Gaussian processes
3 Latent force models
4 Spatio-temporal Gaussian processes (i.e., fields)
5 Summary
Gaussian process regression:

- GPs are used as non-parametric prior models for "learning" input-output $\mathbb{R}^d \rightarrow \mathbb{R}^m$ mappings in form $y = f(x)$.
- A set of noisy training samples $D = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ given.
- The values of function $f(x)$ at measurement points and test points are of interest.

Gaussian process (GP) or Gaussian field is a random function $f(x)$, such that all finite-dimensional distributions $p(f(x_1), \ldots, f(x_n))$ are Gaussian.

- Note that $x$ is the input – not the state! And $f(\bullet)$ is not the drift! – BEWARE of the notation!
- GP can be defined in terms of mean and covariance functions:

$$m(x) = \mathbb{E}[f(x)]$$
$$K(x, x') = \mathbb{E}[(f(x) - m(x))(f(x') - m(x'))^T].$$
The joint distribution of an arbitrary collection of random variables \( f(x_1), \ldots, f(x_n) \) is then given as

\[
\begin{pmatrix}
  f(x_1) \\
  \vdots \\
  f(x_n)
\end{pmatrix}
\sim N
\begin{pmatrix}
  m(x_1) \\
  \vdots \\
  m(x_n)
\end{pmatrix},
\begin{pmatrix}
  K(x_1, x_1) & \cdots & K(x_1, x_n) \\
  \vdots & \ddots & \vdots \\
  K(x_n, x_1) & \cdots & K(x_n, x_n)
\end{pmatrix}
\]

**Temporal Gaussian process** (GP) is a temporal random function \( f(t) \), such that joint distribution of \( f(t_1), \ldots, f(t_n) \) is always Gaussian.

Note that on this course we have denoted these as \( x(t) \!\). In this case the input is the time \( t \) and thus our regressor functions have the form \( y = f(t) \).

Mean and covariance functions have the form:

\[
m(t) = E[f(t)]
\]

\[
K(t, t') = E[(f(t) - m(t))(f(t') - m(t'))^T].
\]
Gaussian Process Regression [1/5]

- Gaussian process regression considers predicting the value of an unknown function ($y$ and $x$ are scalar for illustration)

  $y = f(x)$

at a certain test point ($y^*, x^*$) based on a finite number of training samples ($y_j, x_j$) observed from it.

- To keep the notation less confusing, let’s replace $x$ with $t$:

  $y = f(t)$.

- In classic regression, we postulates parametric form of $f(t; \theta)$ and estimate the parameters $\theta$.

- In GP regression, we instead assume that $f(t)$ is a sample from a Gaussian process with a given covariance function $K(t, t')$, e.g.,

  $K(t, t') = s^2 \exp \left( -\frac{1}{2\ell^2} ||t - t'||^2 \right)$,
Gaussian Process Regression [2/5]

- Let’s denote the vector of observed points as \( \mathbf{y} = (y_1, \ldots, y_n) \), and test point as \( y^* \).
- **Gaussian process assumption** implies that their joint distribution is

\[
\begin{pmatrix}
\mathbf{y} \\
y^*
\end{pmatrix} \sim \mathcal{N}
\begin{pmatrix}
\mathbf{0} \\
\mathbf{0}
\end{pmatrix},
\begin{pmatrix}
\mathbf{K}(t_1:m, t_1:m) & \mathbf{K}^T(t^*, t_1:m) \\
\mathbf{K}(t^*, t_1:m) & \mathbf{K}(t^*, t^*)
\end{pmatrix}
\]

where
- \( \mathbf{K}(t_1:m, t_1:m) = [K(t_i, t_j)] \) is the joint covariance of observed points,
- \( K(t^*, t^*) \) is the (co)variance of the test point,
- \( \mathbf{K}(t^*, t_1:m) = [K(t^*, t_j)] \) is the cross covariance.

By using the **computation rules of Gaussian distributions** we get

\[
\mathbb{E}[y^* | \mathbf{y}] = \mathbf{K}(t^*, t_1:m) \mathbf{K}^{-1}(t_1:m, t_1:m) \mathbf{y}
\]

\[
\text{Var}[y^* | \mathbf{y}] = K(t^*, t^*) - \mathbf{K}(t^*, t_1:m) \mathbf{K}^{-1}(t_1:m, t_1:m) \mathbf{K}^T(t^*, t_1:m).
\]

- These equations can be used for **interpolating** the value of \( y^* = f(t^*) \) at any test point \( t^* \).
In practice, the measurements usually have noise:

\[ y_k = f(t_k) + e_k, \quad e_k \sim N(0, \sigma^2). \]

We want to estimate the value of the "clean" function \( f(t^*) \) at a test point \( t^* \).

Due to the Gaussian process assumption we now get

\[
\begin{pmatrix} y \\ f(t^*) \end{pmatrix} = \mathcal{N} \left( \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} K(t_1:m, t_1:m) + \sigma^2 I & K^T(t^*, t_1:m) \\ K(t^*, t_1:m) & K(t^*, t^*) \end{pmatrix} \right)
\]

The conditional mean and variance are given as

\[
E[f(t^*) | y] = K(t^*, t_1:m) (K(t_1:m, t_1:m) + \sigma^2 I)^{-1} y
\]

\[
\text{Var}[f(t^*) | y] = K(t^*, t^*) - K(t^*, t_1:m) (K(t_1:m, t_1:m) + \sigma^2 I)^{-1} K^T(t^*, t_1:m).
\]

These are the Gaussian process regression equations in their typical form - scalar special cases though.
Gaussian Process Regression [4/5]
The GP-regression has cubic computational complexity $O(m^3)$ in the number of measurements.

This results from the inversion of the $m \times m$ matrix:

$$\mathbb{E}[f(t^*) \mid y] = K(t^*, t_1:m) (K(t_1:m, t_1:m) + \sigma^2 I)^{-1} y$$

$$\text{Var}[f(t^*) \mid y] = K(t^*, t^*) - K(t^*, t_1:m) (K(t_1:m, t_1:m) + \sigma^2 I)^{-1} K^T(t^*, t_1:m).$$

In practice, we use Cholesky factorization and do not invert explicitly – but still the $O(m^3)$ problem remains.

Various sparse, reduced-rank, and related approximations have been developed for this purpose.

Here we study another method – we reduce GP regression into Kalman filtering/smoothing problem which has linear $O(m)$ complexity – in time direction.
Representations of Temporal Gaussian Processes

Sample functions

Covariance function

Spectral density

\[ C(\tau) \]

\[ S(\omega) \]

Simo Särkkä (Aalto)
Example: Ornstein-Uhlenbeck process – path representation as a stochastic differential equation (SDE):

\[
\frac{df(t)}{dt} = -\lambda f(t) + w(t).
\]

where \( w(t) \) is a white noise process.

The mean and covariance functions:

\[
m(t) = 0
\]

\[
k(t, t') = \exp(-\lambda |t - t'|)
\]

Spectral density:

\[
S(\omega) = \frac{2\lambda}{\omega^2 + \lambda^2}
\]

Ornstein-Uhlenbeck process \( f(t) \) is Markovian in the sense that given \( f(t) \) the past \( \{f(s), s < t\} \) does not affect the distribution of the future \( \{f(s'), s' > t\} \).
Consider a $N$th order LTI SDE of the form

$$
\frac{d^N f}{dt^N} + a_{N-1} \frac{d^{N-1} f}{dt^{N-1}} + \cdots + a_0 f = w(t).
$$

If we define $f = (f, \ldots, d^{N-1} f / dt^{N-1})$, we get a state space model:

$$
\begin{align*}
\frac{df}{dt} &= \begin{pmatrix} 0 & 1 \\
& \ddots \\
& & 0 & 1 \\
-a_0 & -a_1 & \cdots & -a_{N-1} \end{pmatrix} f + \begin{pmatrix} 0 \\
\vdots \\
0 \\
1 \end{pmatrix} w(t) \\
\begin{align*}
f(t) &= \begin{pmatrix} 1 & 0 & \cdots & 0 \end{pmatrix} f.
\end{align*}
\end{align*}
$$

The vector process $f(t)$ is now time-Markovian although $f(t)$ is not.
By taking the Fourier transform of the LTI SDE, we can derive the spectral density which has the form:

\[ S(\omega) = \frac{\text{constant}}{(\text{polynomial in } \omega^2)} \]

It turns out that we can also do this conversion to the other direction:

- With certain parameter values, the Matérn has the form:
  \[ S(\omega) \propto (\lambda^2 + \omega^2)^{-(p+1)}. \]

- Many non-rational spectral densities can be approximated, e.g.:
  \[ S(\omega) = \sigma^2 \sqrt{\frac{\pi}{\kappa}} \exp\left(-\frac{\omega^2}{4\kappa}\right) \approx \frac{\text{(const)}}{N!/0!(4\kappa)^N + \cdots + \omega^{2N}} \]

For the conversion of a rational spectral density to a Markovian (state-space) model, we can use the classical spectral factorization –
Spectral factorization finds rational stable transfer function

\[
G(i\omega) = \frac{b_m (i\omega)^M + \cdots + b_1 (i\omega) + b_0}{(i\omega)^N + \cdots + a_1 (i\omega) + a_0}
\]

such that

\[
S(\omega) = G(i\omega) q_c G(-i\omega).
\]

The procedure practice:

- Compute the roots of the numerator and denominator polynomials.
- Construct the numerator and denominator polynomials of the transfer function \(G(i\omega)\) from the positive-imaginary-part roots only.

The SDE is then the inverse Fourier transform of

\[
F(i\omega) = G(i\omega) W(i\omega).
\]

Can be further converted into a state space model –
We have a Fourier-domain system with white noise input:

\[ F(i\omega) = \left( \frac{b_M (i\omega)^M + \cdots + b_1 (i\omega) + b_0}{(i\omega)^N + \cdots + a_1 (i\omega) + a_0} \right) W(i\omega). \]

A standard conversion from transfer function form into state-space form (see control engineering literature), e.g.,

\[
\begin{align*}
\frac{df}{dt} &= \begin{pmatrix}
0 & 1 \\
\vdots & \ddots & \ddots \\
0 & \cdots & 1 \\
-a_0 & -a_1 & \cdots & -a_{N-1}
\end{pmatrix} f + \begin{pmatrix}
0 \\
\vdots \\
0 \\
1
\end{pmatrix} w(t).
\end{align*}
\]

\[ f(t) = \begin{pmatrix} b_0 & b_1 & \cdots & b_M & 0 & \cdots & 0 \end{pmatrix} f. \]
Consider a Gaussian process regression problem of the form
\[ f(x) \sim \mathcal{GP}(0, k(x, x')) \]
\[ y_i = f(x_i) + e_i, \quad e_i \sim \mathcal{N}(0, \sigma_{\text{noise}}^2). \]

Renaming \( x \) into time \( t \) gives:
\[ f(t) \sim \mathcal{GP}(0, k(t, t')) \]
\[ y_i = f(t_i) + e_i, \quad e_i \sim \mathcal{N}(0, \sigma_{\text{noise}}^2). \]

We can now convert this to a state estimation problem:
\[ \frac{df(t)}{dt} = F f(t) + L w(t) \]
\[ y_i = H f(t_i) + e_i. \]

The GP-regression solution \( p(f(t^*) \mid y_1, \ldots, y_m) \) can now be computed with Kalman filter and RTS smoother!
Example (1D Matérn covariance function)

1D Matérn family is ($\tau = |t - t'|$):

$$k(\tau) = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left( \sqrt{2\nu} \frac{\tau}{l} \right)^\nu K_\nu \left( \sqrt{2\nu} \frac{\tau}{l} \right),$$

where $\nu, \sigma, l > 0$ are the smoothness, magnitude and length scale parameters, and $K_\nu(\cdot)$ the modified Bessel function.

For example, when $\nu = 5/2$, we get

$$\frac{d\mathbf{f}(t)}{dt} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -\lambda^3 & -3\lambda^2 & -3\lambda \end{pmatrix} \mathbf{f}(t) + \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} w(t).$$
Inference in Practice

- **Conventional** GP regression:
  1. Evaluate the covariance function at the training and test set points.
  2. Use GP regression formulas to compute the posterior process statistics.
  3. Use the mean function as the prediction.

- **State-space** GP regression:
  1. Form the state space model.
  2. Run Kalman filter through the measurement sequence.
  3. Run RTS smoother through the filter results.
  4. Use the smoother mean function as the prediction.

With both GP regression and state-space formulation we have the corresponding parameter estimation methods – see, e.g., Rasmussen & Williams (2006) and Särkkä (2013), respectively.
State-Space GP Regression Demo
Comparison of GP regression (L) and RTS smoother (R) results
A latent force model (LFM) is of the form
\[
\frac{dx_f(t)}{dt} = g(x_f(t)) + u(t),
\]
where \(u(t)\) is the latent force.

We measure the system at discrete instants of time:
\[
y_k = x_f(t_k) + r_k
\]

Let’s now model \(u(t)\) as a Gaussian process of Matern type
\[
K(\tau) = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left( \sqrt{2\nu} \frac{\tau}{l} \right)^\nu K_\nu \left( \sqrt{2\nu} \frac{\tau}{l} \right)
\]
Recall that if, for example, \(\nu = 1/2\) then the GP can be expressed as the solution of the stochastic differential equation (SDE)
\[
\frac{du(t)}{dt} = -\lambda u(t) + w(t)
\]
If we define $x = (x_f, u)$, we get a two-dimensional SDE

$$\frac{dx}{dt} = \begin{pmatrix} g(x_1(t)) + x_2(t) \\ -\lambda x_2(t) \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} w(t)$$

We can now rewrite the measurement model as

$$y_k = \begin{pmatrix} 1 & 0 \end{pmatrix} x(t_k) + r_k$$

Thus the result is a model of the generic form

$$\frac{dx}{dt} = a(x) + L \, w(t)$$

$$y_k = H \, x(t_k) + r_k.$$ 

This model can now be efficiently tackled with non-linear Kalman filtering and smoothing.
The dynamics of the car in 2d $(x_1, x_2)$ are given by Newton’s law:

$$F(t) = ma(t),$$

where $a(t)$ is the acceleration, $m$ is the mass of the car, and $F(t)$ is a vector of (unknown) forces acting the car.

We model $F(t)/m$ as a 2-dimensional white noise process:

$$d^2 x_1 / dt^2 = w_1(t)$$
$$d^2 x_2 / dt^2 = w_2(t).$$
If we define $x_3(t) = dx_1/dt$, $x_4(t) = dx_2/dt$, then the model can be written as a first order system of differential equations:

$$
\frac{d}{dt} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \end{pmatrix}.
$$

In shorter matrix form:

$$
\frac{d\mathbf{x}}{dt} = \mathbf{F} \mathbf{x} + \mathbf{L} \mathbf{w}.
$$
Assume that the position of the car \((x_1, x_2)\) is measured and the measurements are corrupted by Gaussian measurement noise \(e_{1,k}, e_{2,k}\):

\[
y_{1,k} = x_1(t_k) + e_{1,k} \\
y_{2,k} = x_2(t_k) + e_{2,k}.
\]

The measurement model can be now written as

\[
y_k = H x(t_k) + e_k, \quad H = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}
\]
The dynamic and measurement models of the car now form a linear Gaussian state-space model:

\[
\frac{dx}{dt} = Fx + Lw \\
y_k = Hx(t_k) + r_k,
\]

In this case it is possible to solve the transition density explicitly:

\[
p(x(t_k) \mid x(t_{k-1})) = N(x(t_k) \mid A_{k-1}x(t_{k-1}), Q_{k-1})
\]

where \( A_{k-1} \) and \( Q_{k-1} \) can be expressed in terms of the matrix exponential function (see yesterday’s lecture).

Thus we can actually write this as a discrete-time model:

\[
x_k = A_{k-1}x_{k-1} + q_{k-1} \\
y_k = Hx_k + r_k,
\]

where \( q_{k-1} \sim N(0, Q_{k-1}) \).
We can also start from a latent force model

\[
\frac{d^2 x_1}{dt^2} = u(t) \\
\frac{d^2 x_2}{dt^2} = v(t),
\]

where \( u \) and \( v \) are, say, Matern 3/2 processes.

In state-space form this leads to

\[
\frac{d\mathbf{u}(t)}{dt} = \begin{pmatrix} 0 & 1 \\ -\lambda^2 & -2\lambda \end{pmatrix} \mathbf{u}(t) + \begin{pmatrix} 0 \\ 1 \end{pmatrix} w_u(t), \quad j = 1, 2
\]

where \( \mathbf{u}(t) = (u(t), du(t)/dt) \).

We can also have both white noises and latent forces:

\[
\frac{d^2 x_1}{dt^2} = u(t) + w_1(t) \\
\frac{d^2 x_2}{dt^2} = v(t) + w_2(t).
\]
Now we get

\[
\frac{d}{dt} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ u_1 \\ v_1 \\ u_2 \\ v_2 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & -\lambda^2 & 0 & -2\lambda & 0 \\ 0 & 0 & 0 & 0 & -\lambda^2 & 0 & -2\lambda & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ u_1 \\ u_2 \\ v_1 \\ v_2 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \\ w_u \\ w_v \end{pmatrix}
\]

\[ y_k = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ u_1 \\ u_2 \\ v_1 \\ v_2 \end{pmatrix} + e_k, \]

But this is just a **linear Gaussian state-space model**:

\[
\frac{dx}{dt} = Fx + Lw
\]

\[ y_k = Hx(t_k) + r_k, \]
The temporal vector-valued process becomes an infinite-dimensional function (Hilbert space) -valued process:

\[ f(t) = \begin{pmatrix} f_1(t) \\ \vdots \\ f_n(t) \end{pmatrix} \rightarrow \begin{pmatrix} f(x_1, t) \\ \vdots \\ f(x_n, t) \end{pmatrix} \rightarrow f(x, t), \quad x \in \mathbb{R}^d. \]
Representations of Spatio-Temporal Gaussian Processes

- **Moment representation** in terms of mean and covariance function

  \[ m(x, t) = \mathbb{E}[f(x, t)] \]
  \[ K(x, x'; t, t') = \mathbb{E}[(f(x, t) - m(x, t))(f(x', t') - m(x', t'))^T]. \]

- **Spectral representation** in terms of spectral density function

  \[ S(\omega_x, \omega_t) = \tilde{f}(i \omega_x, i \omega_t)\tilde{f}^T(-i \omega_x, -i \omega_t). \]

- As an infinite-dimensional state space model or stochastic evolution equation:

  \[ \frac{\partial f(x, t)}{\partial t} = \mathcal{F} f(x, t) + L w(x, t). \]
Infinite-dimensional state-space model with operators $\mathcal{F}$ and $\mathcal{H}_i$:

$$\frac{\partial f(x, t)}{\partial t} = \mathcal{F} f(x, t) + L w(x, t)$$

$$y_i = \mathcal{H}_i f(x, t_i) + e_i$$

We can use the infinite-dimensional Kalman filter and RTS smoother – scale linearly in time dimension.

We can approximate with PDE methods such as basis function expansions, FEM, finite-differences, spectral methods, etc.

If $\mathcal{F}$ and $\mathcal{H}_i$ are “diagonal” in the sense that they only involve point-wise evaluation in $x$, we get a finite-dimensional algorithm.

- Diagonal $\mathcal{F}$ corresponds to a separable model.
- The evaluation operator $\mathcal{H}_i$ in GP regression and Kriging is diagonal.
We can convert spatio-temporal covariance functions into state-space models as follows:

1. First compute the spectral density $S(\omega_x, \omega_t)$ by Fourier transforming the covariance function $K(x, t)$.
2. Form rational approximation in variable $i\omega_t$:
   
   $$S(\omega_x, \omega_t) = \frac{q(i\omega_x)}{b_0(i\omega_x) + b_1(i\omega_x)(i\omega_t) + \cdots + (i\omega_t)^N}.$$

3. Form the corresponding Fourier domain SDE (via the spectral factorization again):
   
   $$\frac{\partial^N \tilde{f}(\omega_x, t)}{\partial t^N} + a_{N-1}(i\omega_x) \frac{\partial^{N-1} \tilde{f}(\omega_x, t)}{\partial t^{N-1}} + \cdots + a_0(i\omega_x) \tilde{f}(\omega_x, t) = \tilde{w}(\omega_x, t).$$
Conversion of Spatio-Temporal Covariance into Infinite-Dimensional State Space Model (cont.)

... conversion method continues ...

By converting this to state space form and by taking spatial inverse Fourier transform, we get the stochastic evolution equation

\[
\frac{\partial \mathbf{f}(\mathbf{x}, t)}{\partial t} = \mathcal{F} \begin{pmatrix}
0 & 1 & & \\
& \ddots & \ddots & \\
& & 0 & 1 \\
-\mathcal{A}_0 & -\mathcal{A}_1 & \cdots & -\mathcal{A}_{N-1}
\end{pmatrix} \mathbf{f}(\mathbf{x}, t) + \begin{pmatrix}
0 \\
\vdots \\
0 \\
1
\end{pmatrix} \mathbf{w}(\mathbf{x}, t)
\]

where \( \mathcal{A}_j \) are pseudo-differential operators.

We can now use infinite-dimensional Kalman filter and RTS smoother for efficient estimation of the “state” \( \mathbf{f}(\cdot, t) \).
Example (2D Matérn covariance function)

- The multidimensional Matérn covariance function is the following:
  \( r = \|\xi - \xi'\| \), for \( \xi = (x_1, x_2, \ldots, x_{d-1}, t) \in \mathbb{R}^d \):

  \[
  k(r) = \sigma^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left( \sqrt{2\nu} \frac{r}{l} \right)^\nu K_\nu \left( \sqrt{2\nu} \frac{r}{l} \right).
  \]

- For example, if \( \nu = 1 \) and \( d = 2 \), we get the following:

  \[
  \frac{\partial f(x, t)}{\partial t} = \begin{pmatrix} 0 & 1 \\ \frac{\partial^2}{\partial x^2} - \lambda^2 & -2\sqrt{\lambda^2 - \frac{\partial^2}{\partial x^2}} \end{pmatrix} f(x, t) + \begin{pmatrix} 0 \\ 1 \end{pmatrix} w(x, t).
  \]
In Gaussian process (GP) regression we put a Gaussian process prior on the regressor functions \( f(t) \).

The value of the function \( f(t^*) \) at a test point \( t^* \) is predicted by conditioning the process on the training set.

GP regression has a problematic cubic \( O(m^3) \) complexity in the number of measurements \( m \).

We can often convert a GP regression problem into a Kalman filtering/smoothing problem which has linear \( O(m) \) time complexity.

But – this is possible only in time-direction.

Latent force models with GP forces can also be converted into Kalman filtering/smoothing problems.

In space-time models we need to use infinite-dimensional Kalman filters and smoothers.