Introduction to recursive machine learning

Modeling

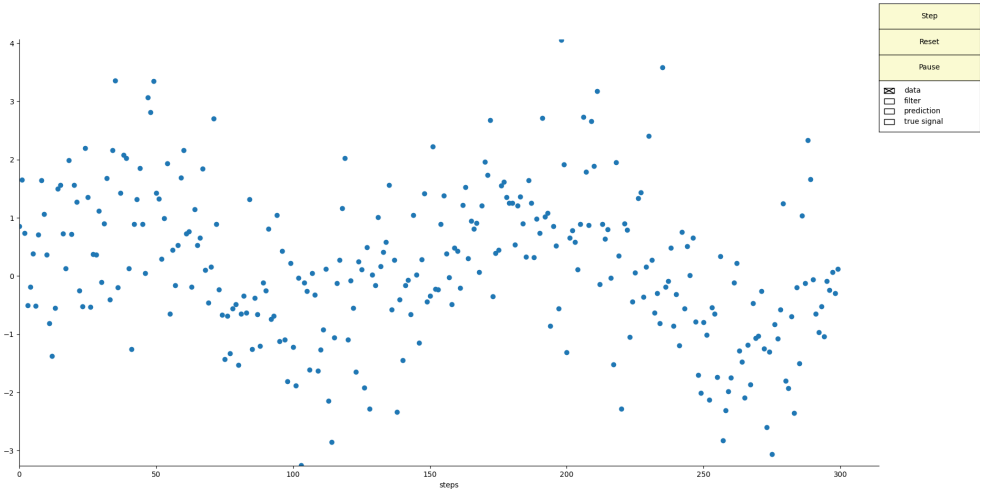
Juan Pablo Carbajal
juanpablo.carbajal@ost.ch

Eastern Switzerland University of Applied Sciences OST
Institute for Energy Technology IET
Scientific Computing and Engineering Group SCE

January 2023 - Rapperswil

Overview

Our goal



source: *s_filtering_interactive.py*

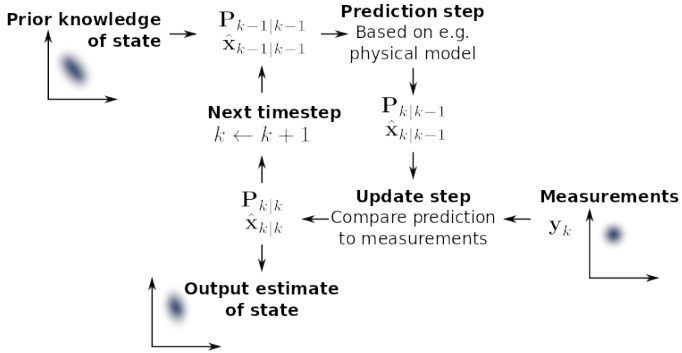
The goal of the workshop is that you are able to understand and configure Kalman filters. You should be able ot understand when to use it, how to use it, and what to do in case it is not applicable.

The needed ingredients

- An initial state $\rightarrow$ **Prior** distribution
- Model for the dynamics of the inferred states: how the states change over time $\rightarrow$ Linear Iterated maps
- Model for how the measurements are obtained from the states $\rightarrow$ Linear Map, **likelihood**
- A method to update the inferred states $\rightarrow$ Bayes rule, **posterior** distribution

Expert knowledge is realized in the models of the dynamics and the measurements.
Model dynamics (structure and parameter values) can also be learned from data (*out fo scope*: data driven dynamical systems, system identification).
Given the model structure parameters can be tuned to the data (*within scope*: parameter estimation)

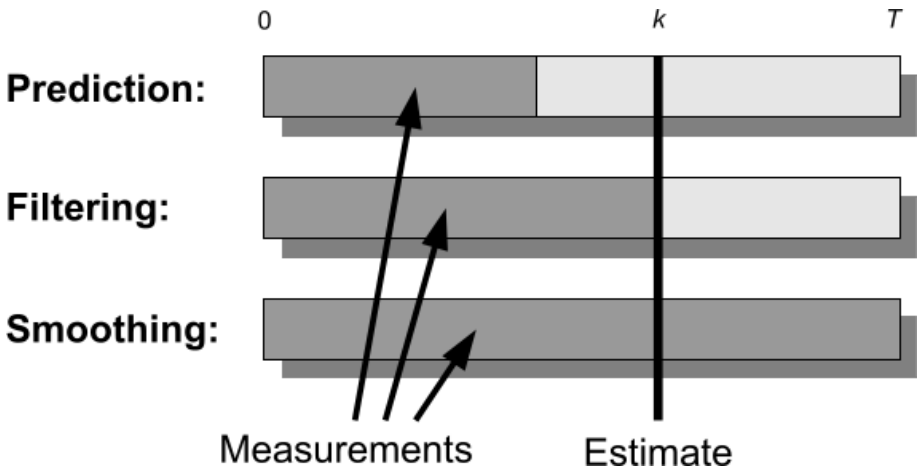
The needed ingredients



source: *Wikipedia, Kalman filter*

Our goal of today is to understand the **prediction step**. In the following session we will cover the **update step**. Finally we will work several examples.

Filtering and smoothing

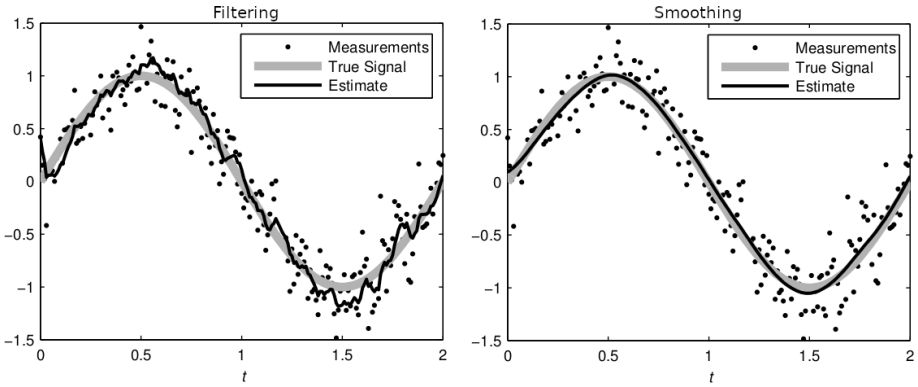


source: *Särkkä, S. doi:10.1017/CBO9781139344203*

State estimation problems can be divided into optimal prediction, filtering, and smoothing depending on the time span of the measurements available with respect to the time of the estimated state:

- **Prediction**: estimate future state, beyond the current measurements and state.
- **Filtering**: estimate current state, given the previous and current measurements.
- **Smoothing**: estimate current state, given previous, current, and future measurements.

Filtering and smoothing



source: *Särkkä, S. doi:10.1017/CBO9781139344203*

Smoothing uses future information to update the past, hence the estimation is smoother.

Let's start with prediction.

Iterated maps

For the prediction step we need to set up a model that allows us to update the state iteratively. A widely used class of models to do this are Ordinary Differential Equations (ODE), which are continuous models. In the seminar we will look at discrete models that update the state in discrete steps. One can obtain discrete models from an ODE, via discretization, we will see that later.
The prediction step then requires a model of the form

$$x_{k+1} = f(x_k, \ldots)$$

where the function f takes the state at step k (any other information) and generates the (predicted) state at step $k+1$.
We look into linear update functions.

Iterated map: single variable

Map (mapping, transformation, etc.):

$$x = \alpha y$$

What do we get if $\alpha = 1$?

Iterated map: single variable

Map (mapping, transformation, etc.):

$$x = \alpha y$$

Iterated map (use the result as the next value of x):

$$x_k = \alpha x_{k-1} \quad |\alpha| \leq 1$$

Do you expect a straight trajectory?
What do we get if $\alpha = 1$ and $\alpha = -1$?

Iterated map: single variable

Map (mapping, transformation, etc.):

$$x = \alpha y$$

Iterated map (use the result as the next value of x):

$$x_k = \alpha x_{k-1} \quad |\alpha| \leq 1$$

0. x_0

We will now iterate the map on the initial condition x_0.

Iterated map: single variable

Map (mapping, transformation, etc.):

$$x = \alpha y$$

Iterated map (use the result as the next value of x):

$$x_k = \alpha x_{k-1} \quad |\alpha| \leq 1$$

0. x_0
1. $x_1 = \alpha x_0$

Iterated map: single variable

Map (mapping, transformation, etc.):

$$x = \alpha y$$

Iterated map (use the result as the next value of x):

$$x_k = \alpha x_{k-1} \quad |\alpha| \leq 1$$

0. x_0
1. $x_1 = \alpha x_0$
2. $x_2 = \alpha x_1 = \alpha\alpha x_0 = \alpha^2 x_0$

Iterated map: single variable

Map (mapping, transformation, etc.):

$$x = \alpha y$$

Iterated map (use the result as the next value of x):

$$x_k = \alpha x_{k-1} \quad |\alpha| \leq 1$$

0. x_0
1. $x_1 = \alpha x_0$
2. $x_2 = \alpha x_1 = \alpha\alpha x_0 = \alpha^2 x_0$
3. $x_3 = \alpha x_2 = \alpha\alpha^2 x_0 = \alpha^3 x_0$

Iterated map: single variable

Map (mapping, transformation, etc.):

$$x = \alpha y$$

Iterated map (use the result as the next value of x):

$$x_k = \alpha x_{k-1} \quad |\alpha| \leq 1$$

0. x_0
1. $x_1 = \alpha x_0$
2. $x_2 = \alpha x_1 = \alpha\alpha x_0 = \alpha^2 x_0$
3. $x_3 = \alpha x_2 = \alpha\alpha^2 x_0 = \alpha^3 x_0$
4. $\ldots$

Iterated map: single variable

Map (mapping, transformation, etc.):

$$x = \alpha y$$

Iterated map (use the result as the next value of x):

$$x_k = \alpha x_{k-1} \quad |\alpha| \leq 1$$

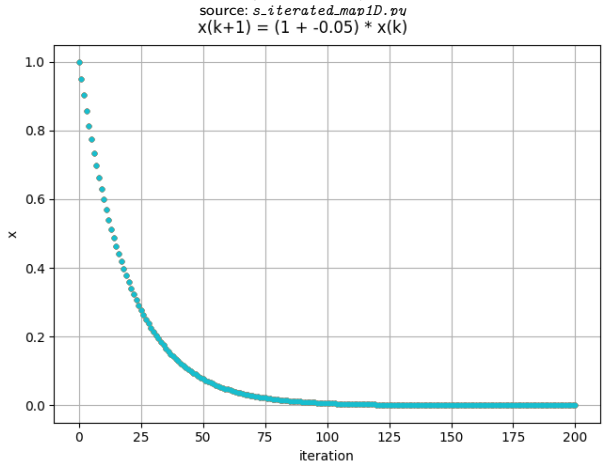
0. x_0
1. $x_1 = \alpha x_0$
2. $x_2 = \alpha x_1 = \alpha\alpha x_0 = \alpha^2 x_0$
3. $x_3 = \alpha x_2 = \alpha\alpha^2 x_0 = \alpha^3 x_0$
4. $\ldots$
5. $x_k = \alpha^k x_0$

The composition of linear maps gives a different linear map at each step: not a straight trajectory

Iterated map: single variable

Example

$$x_k = (1 - 0.05)x_{k-1}$$



Observation: linear map but not straight trajectory

Test other values of $\alpha \in [-1, 1]$. For what values of α do you see qualitative change in behavior?

What happens when $\alpha < -1$ or $\alpha > 1$?

Iterated map: two variables

Matrices and vectors

Transform two variables x_1, x_2 into y_1, y_2

$$y_1 = a_{11}x_1 + a_{12}x_2$$
$$y_2 = a_{21}x_1 + a_{22}x_2$$

The color of the indices suggest a way to order the equations. Do you have any idea?

Iterated map: two variables

Matrices and vectors

Transform two variables x_1, x_2 into y_1, y_2

$$y_1 = a_{11}x_1 + a_{12}x_2$$
$$y_2 = a_{21}x_1 + a_{22}x_2$$

Organize the values in tables (matrices)

matrix: $\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$

single column: $\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ $\boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$

single row: $\boldsymbol{x}^\top = \begin{bmatrix} x_1 & x_2 \end{bmatrix}$

Define the product of rows and columns

$$\begin{bmatrix} x & y \end{bmatrix} \cdot \begin{bmatrix} u \\ w \end{bmatrix} = xu + yw$$

The shape (a.k.a. size) of a matrix with n rows and m columns is written $n \times m$. One can also specify the set to which the entries (or cells) of the matrix belong to, e.g. $\mathbb{R}^{2\times3}$ are matrices with 2 rows and 3 columns with entries in the real numbers. Other examples would be $\mathbb{Z}^{n\times n}$, $\{0,1\}^{n\times m}$, etc.

If $\boldsymbol{x}$ is a column vector, what gives $\boldsymbol{x}\boldsymbol{x}^\top$?
Work this out, because it will help with the definition of the variance of a random vector.

Iterated map: two variables

Matrices and vectors

Transform two variables x_1, x_2 into y_1, y_2

$$y_1 = a_{11}x_1 + a_{12}x_2$$
$$y_2 = a_{21}x_1 + a_{22}x_2$$

Organize the values in tables (matrices)

matrix: $\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$

single column: $\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$ $\boldsymbol{y} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$

single row: $\boldsymbol{x}^\top = \begin{bmatrix} x_1 & x_2 \end{bmatrix}$

Define the product of rows and columns

$$\begin{bmatrix} x & y \end{bmatrix} \cdot \begin{bmatrix} u \\ w \end{bmatrix} = xu + yw$$

The transformation of the variables now looks like

$$\boldsymbol{y} = \mathbf{A}\boldsymbol{x}$$

Re-write the transformation using row vectors.

Iterated map: two variables

Map (mapping, transformation, etc.)

$$\boldsymbol{y} = \mathbf{A}\boldsymbol{x}$$

Iterated map:

$$\boldsymbol{x}_k = \mathbf{A}\boldsymbol{x}_{k-1}$$

$$\mathbf{A} = \mathbf{I} + \alpha \begin{bmatrix} 0 & 1 \\ -1 & -2\alpha \end{bmatrix} \quad 0 \leq \alpha \leq 1$$

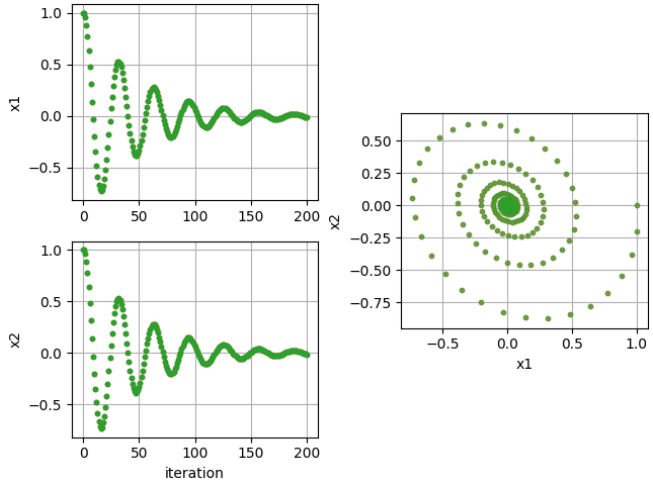
0. $\boldsymbol{x}_0$
1. $\boldsymbol{x}_1 = \mathbf{A}\boldsymbol{x}_0$
2. $\boldsymbol{x}_2 = \mathbf{A}\boldsymbol{x}_1 = \mathbf{A}\mathbf{A}\boldsymbol{x}_0 = \mathbf{A}^2\boldsymbol{x}_0$
3. $\boldsymbol{x}_3 = \mathbf{A}\boldsymbol{x}_2 = \mathbf{A}\mathbf{A}^2\boldsymbol{x}_0 = \mathbf{A}^3\boldsymbol{x}_0$
4. $\ldots$
5. $\boldsymbol{x}_k = \mathbf{A}^k\boldsymbol{x}_0$

The composition of linear maps gives a different linear map at each step: not a straight trajectory

Iterated map: two variables

Example

$$\boldsymbol{x}_k = \left(\mathbf{I} + 0.2 \begin{bmatrix} 0 & 1 \\ -1 & -0.4 \end{bmatrix} \right) \boldsymbol{x}_{k-1}$$



Test other values of $\alpha \in [0, 1]$. For what values of α do you see qualitative changes in behavior?

What happens when $\alpha < 0$ or $\alpha > 1$?

Model of measurements (observations)

Till now we have explore ideas to model the evolution of the states of a system. A system can have many states (components of the vector x), but in practice we do not observe them all. We measure only a few states, or linear mixtures of them, in general we measure any function of the states...

Now we will explore how to model linear measurements.

Measurement model

The states of the model are not necessarily observed. The modelled measurements, the values given by sensors, are obtained from the state of the model:

$$\boldsymbol{x}_k = \mathbf{A}\boldsymbol{x}_{k-1}$$
$$\boldsymbol{y}_k = \mathbf{H}\boldsymbol{x}_k \quad \text{Measurement model}$$

The dimension of $\boldsymbol{y}$ is defined by the number of sensors (or measurement channels). For m measurements and a n dimensional state we have that: $\boldsymbol{x} \in \mathbb{R}^{n\times 1}$, $\boldsymbol{y} \in \mathbb{R}^{m\times 1}$, $\mathbf{A} \in \mathbb{R}^{n\times n}$, $\mathbf{H} \in \mathbb{R}^{m\times n}$.

In general, the measurement model looks like:

$$y_k = h(x_k, \ldots)$$

it provides a model of the measurements as a function of the current state of the model. Herein we focus on linear measurement models.

It is common to have m (number of measurements) lower than n. Hence $\mathbf{H}$ is usually a wider than taller (short-fat matrix).

Modify `s_iterated_map2D.py` (or implement your own) and implement different measurement matrices $\mathbf{H}$. Plot the measurements.

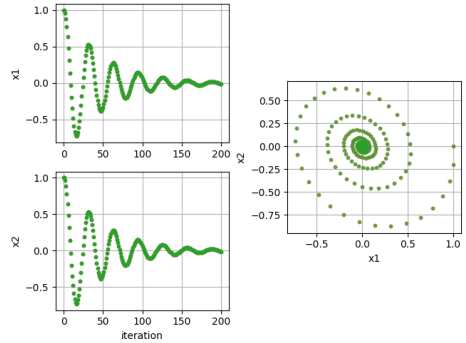
Measurement model

Consider the model:

$$\boldsymbol{x}_k = \begin{bmatrix} 1 & 0.2 \\ -0.2 & 0.92 \end{bmatrix} \boldsymbol{x}_{k-1}$$
$$\boldsymbol{y}_k = \mathbf{H}\boldsymbol{x}_k$$

Measure 1st component:
$\mathbf{H} = \begin{bmatrix} 1 & 0 \end{bmatrix}$
Measure 2nd component:
$\mathbf{H} = \begin{bmatrix} 0 & 1 \end{bmatrix}$
What does this measures?

$$\mathbf{H} = \begin{bmatrix} -1 & 1 \end{bmatrix}$$



1. What's $\mathbf{H}$ if we measure the mean of the two components?
2. What's $\mathbf{H}$ if we measure x_1 and $x_1 + x_2$?

Inputs (deterministic perturbations)

Till now we have seen **autonomous** system models (a.k.a. Linear Time Invariant, LTI). When the evolution of the states is also affected by an external signal (a.k.a. exogenous input), the system is not autonomous anymore, sometimes it is called *input-driven*.

Now we will explore how to model linear inputs.

Inputs

$$\boldsymbol{x}_k = \mathbf{A}\boldsymbol{x}_{k-1} + \mathbf{B}\boldsymbol{u}_{k-1} \quad \text{Input matrix, inputs}$$
$$\boldsymbol{y}_k = \mathbf{H}\boldsymbol{x}_k$$

For n states and k inputs, the input matrix has the shape $n \times k$.
Can you provide an example, even if it is 1-dimensional?

Inputs

$$\boldsymbol{x}_k = \mathbf{A}\boldsymbol{x}_{k-1} + \mathbf{B}\boldsymbol{u}_{k-1} \quad \text{Input matrix, inputs}$$
$$\boldsymbol{y}_k = \mathbf{H}\boldsymbol{x}_k$$

Example from kinematics: the evolution of the position and speed of a point-mass moving in 1D.

$$p_t = p_{t-1} + v_{t-1}\Delta t \qquad v_t = v_{t-1} + \underbrace{a_{t-1}}_{\frac{F_{t-1}}{m}}\Delta t$$

define

$$\boldsymbol{x}_t = \begin{bmatrix} p_t \\ v_t \end{bmatrix} \quad \mathbf{A} = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \quad u_t = a_t \quad \mathbf{B} = \begin{bmatrix} 0 \\ \Delta t \end{bmatrix}$$

then we get (assuming we observe only the position)

$$\boldsymbol{x}_t = \mathbf{A}\boldsymbol{x}_{t-1} + \mathbf{B}u_{t-1}$$
$$y_t = \begin{bmatrix} 1 & 0 \end{bmatrix} \boldsymbol{x}_t$$

The example is the Euler method, which is not a good discretization unless Δt is very small. An alternative map would be

$$p_t = p_{t-1} + v_{t-1}\Delta t + \frac{1}{2}a_{t-1}\Delta t^2$$
$$\mathbf{B} = \begin{bmatrix} \frac{\Delta t^2}{2} \\ \Delta t \end{bmatrix}$$

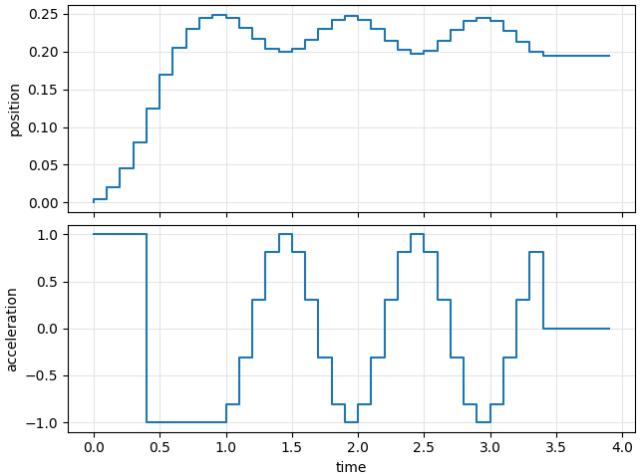
and a_{t-1} is the mean acceleration between $t-1$ and t.
One can also get rid of the velocity using Verlet integration without velocity (Störmer method), which uses central differences for the second derivative:

$$p_t = 2p_{t-1} - p_{t-2} + a_{t-1}\Delta t^2$$

What are $\mathbf{B}$, $\mathbf{H}$, and $\boldsymbol{u}_t$ in this case? How do those change if $a_t = -kp_t$?

Inputs

source: *s_kinematic_map.py*



Examples

Iterated map for linear regression

Given some examples of a line (two would suffice): $\{(t_i, y_i)\}_{i=1,2,\ldots}$. Find the slope and intercept of a straight line that goes through the points:

$$at_i + b = y_i$$

Batch version: build the design matrix (also data matrix, Vandermonde matrix for polynomial regression) and the unknown coefficients vector:

$$D \begin{bmatrix} a \\ b \end{bmatrix} = \boldsymbol{y}, \quad D = \begin{bmatrix} t_1 & 1 \\ \vdots & \vdots \\ t_n & 1 \end{bmatrix} \quad \rightarrow \begin{bmatrix} a \\ b \end{bmatrix} = D^{-1}\boldsymbol{y}$$

Issues: 1) the independent variable might grow to very large numbers; 2) The more data we get, the worst the conditioning of the design matrix.

Not the best choice for an online algorithm!

The design matrix can be built with any feature (function) of the regressor, e.g. polynomial regression would look just like this, alas with a wider Vandermonde matrix.

By solving the problem with, e.g. least squares one recovers the coefficients of the line.
If you never did, write the a program that solves this problem. That is, a function that consumes the data and returns the coefficient of the line.

In the batch approach the more data we get the worst is the conditioning of the design matrix. Also the independent t variable might grow to very large numbers. Hence this solution is not the best choice for an online algorithm.

Examples

Iterated map for linear regression

The **dynamic model for a straight line**: if we get a point already on the line, a neighboring point is obtained by

$$y_k = y_{k-1} + a\Delta t_k \qquad \Delta t_k = t_k - t_{k-1}$$

Δt is not expected to grow indefinitely (unlike t in batch regression). The iteration above, defines the dynamic and measurement model

$$x_k = x_{k-1}, \qquad x_0 = \begin{bmatrix} a \\ \\ y_0 \end{bmatrix}$$

$$y_k = H_k x_k, \qquad H_k = \begin{bmatrix} \Delta t_k & 1 \end{bmatrix}$$

You can get this insight by looking at the two points:

$$y_{k-1} = at_{k-1} + b$$
$$y_k = at_k + b$$

and replacing b in the second equation, using the first one.

1. What is the system matrix $\mathbf{A}$ in this case?
2. What type of dynamics does it model?
3. Is there any (exogenous) inputs? What is the input matrix $\mathbf{B}$
4. What would the state x_k and the measurement matrix be for polynomial regression? $y(t) = a_0 + a_1 t + a_2 t^2 + \dots$
5. What would the state and the measurement matrix be for a linear mixture of functions? $y(t) = \sum_{i=0}^{N} a_i \varphi_i(t)$

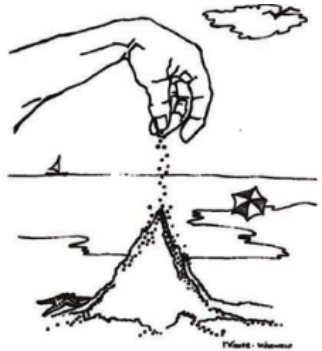
Coffee break: 15 minutes

Error propagation

We will investigate how a small deviations propagate through function. The goal is to build some intuition and to expose ourselves to the structure of the emerging formulas.

Linearly related variables

Take two variables related by: $y = b + ax$

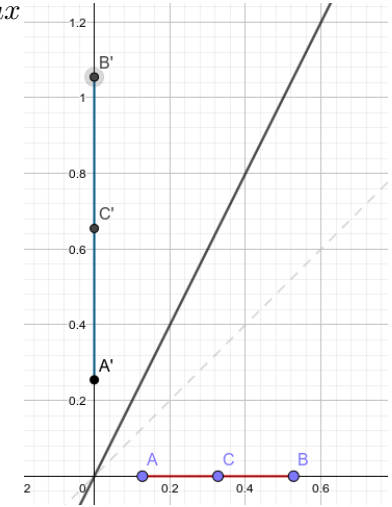


Linearly related variables

Take two variables related by: $y = b + ax$

At a given value x_o we get:
$y_o = b + ax_o$
If we modify x_o by a given amount Δx:

$$\hat{y} = b + a(x_o + \Delta x) = \underbrace{b + ax_o}_{=y_o} + a\Delta x$$
$$= y_o + a\Delta x$$



source: https://www.geogebra.org/m/peude6ek

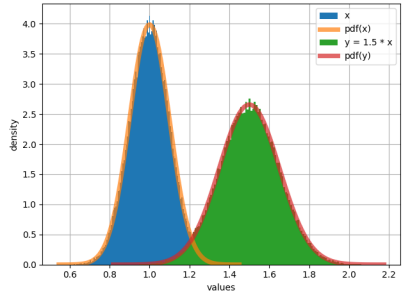
Linearly related variables

Take two variables related by: $y = b + ax$

At a given value x_o we get:
$y_o = b + ax_o$
If we modify x_o by a given amount Δx:

$$\hat{y} = b + a(x_o + \Delta x) = \underbrace{b + ax_o}_{=y_o} + a\Delta x$$
$$= y_o + a\Delta x$$

source: *s_sampling_mapping.py*



Then y_o changes an amount Δy:

$$\hat{y} = y_o + \Delta y$$
$$\Delta y := a\Delta x$$

Linearly related variables

Take two variables related by: $y = b + ax$

$$\Delta y := a\Delta x$$

The "error" Δx propagates to y via the slope a of the relation. The intercept b doesn't play a role in the induced "error" Δy.

Can you propagate and initial value error on an iterated map?

Error propagation in an 1D interated map

$$x_k = a x_{k-1}$$

Using our previous result

$$\Delta x_k = a \Delta x_{k-1}$$

It follows the same dynamics as the state!
Let's propagate an initial value error

0. $x_0 + \Delta x_0$
1. $x_1 = a\,(x_0 + \Delta x_0)$
2. $x_2 = a^2\,(x_0 + \Delta x_0)$
3. $\dots$
4. $x_k = a^k\,(x_0 + \Delta x_0)$

It is just the iterated map on a different initial condition!

Implement these formulas in the computer and run some simulations for the same x_0 and different values of Δx_0.

For many simulations with different values of Δx_0, what's the mean value of x_k?

Open question: what if we made an error at each iteration step?

What changes if the state is multi-dimensional $\boldsymbol{x}_k$? Consider these matrices:

$$\mathbf{A}_d = \frac{1}{2}\mathbf{I} \quad \mathbf{A}_m = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ -\frac{1}{3} & \frac{1}{2} \end{bmatrix}$$

and propagate an error in the 1st component of the initial value , i.e. $\boldsymbol{x}_o^\top = \begin{bmatrix} x_{o1} + \Delta x_{o1} & x_{o2} \end{bmatrix}$

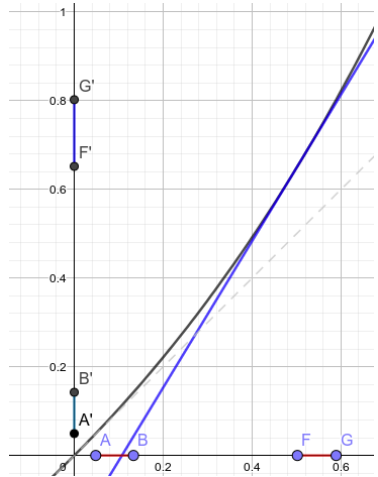
Non-linearly connected variables and small error

Consider two variables connected by a nonlinear relation: $y = f(x)$

At a given value x_o we get $y_o = f(x_o)$

If we modify x_o by a given amount Δx:

$$\hat{y} = f(x_o + \Delta x)$$



source: https://www.geogebra.org/m/vqppqnwz

Non-linearly connected variables and small error

Consider two variables connected by a nonlinear relation: $y = f(x)$

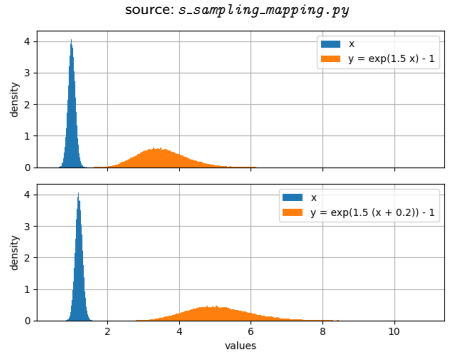
If Δx is small (and f analytic), then we can proceed:

$$\hat{y} = f(x_o + \Delta x) \approx f(x_o) + \frac{\partial f}{\partial x}\Big|_{x_o} \Delta x$$

$$\hat{y} = y_o + \frac{\partial f}{\partial x}\Big|_{x_o} \Delta x = y_o + \Delta y$$

$$\Delta y := \frac{\partial f}{\partial x}\Big|_{x_o} \Delta x$$

source: *s_sampling_mapping.py*



Finding the best linear approximation of a nonlinear function, is called **linearization**.
It corresponds to the truncated Taylor expansion of degree 1. See `https://www.geogebra.org/m/C4S6CEdm` for examples.

1. Write a program to propagate random samples through the exponential function $y(x) = \exp(2x)$.
2. Choose the mean of the random samples as x_o and linearize the exponential function around x_o. Propagate the values using the resulting linear mapping.
3. Increase the variance of the random samples, is the linearization similar to the propagation using the function directly?
4. Try other non-linear functions

Non-linearly connected variables and small error

Consider two variables connected by a nonlinear relation: $y = f(x)$

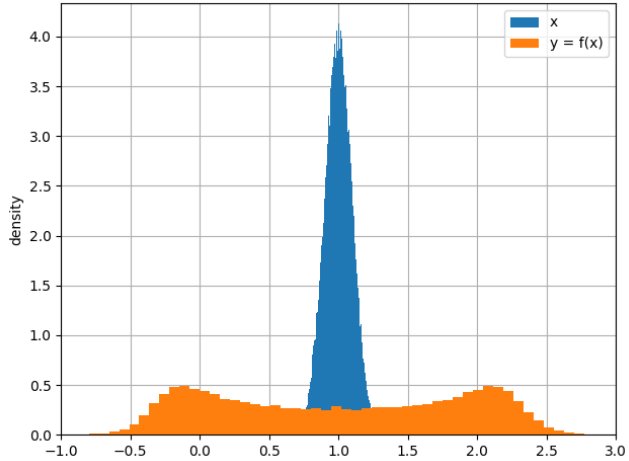
$$\Delta y \coloneqq \frac{\partial f}{\partial x}\Big|_{x_o} \Delta x$$

The small "error" Δx propagates to y via the **local** slope of the relation. For different values of x_o the "error"propagates differently.
There are other approaches for the non-linear case that do not make the same assumptions we did here. This is not on the scope of the course. A good place to start would be Wikipedia's article on propagation of uncertainty
`https://en.wikipedia.org/wiki/Propagation_of_uncertainty`.

Non-linearly connected variables

Nonlinear mappings can radically change the distribution

$$y = \frac{1}{2}\tanh(15(x-\mu))\left(e^{3|x-\mu|}+1\right)+1$$



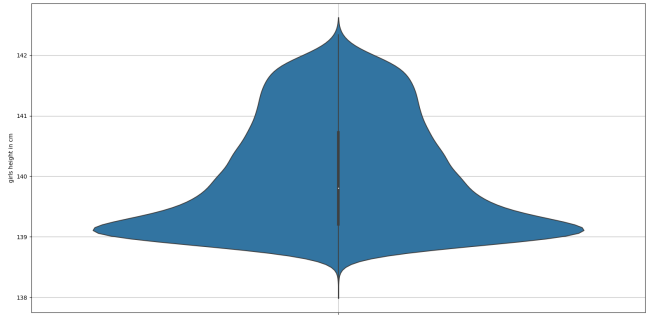
Gaussian (Normal) distribution

We overview properties and visualizations of the Gaussian (Normal) distribution in several dimensions. We review the concepts of mean and variance. The goal is to refresh the concept and to conceptually introduce random processes as an infinite dimensional distribution.

1-dimensional (single variable) distribution

Box or violin plot view

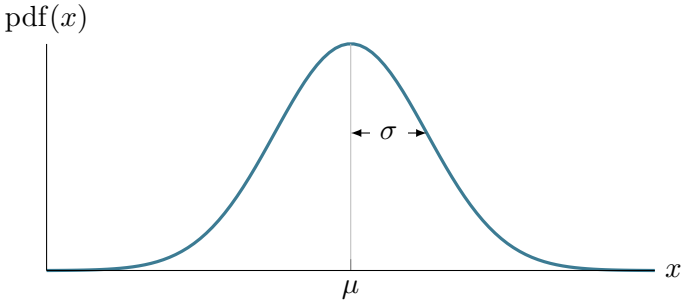
year	height
1985	138.8
1986	139.0
1987	139.0
1988	138.8
⋮	⋮
2010	138.8
2011	138.7
2012	139.2
2013	139.0
2014	139.0
2015	138.6
⋮	⋮



The data is the height of many girls over several years. We can look at the distribution of all these heights making a violin plot. In the middle of the violin you see a box plot. The body of the violin is a smoothed version of an histogram. The body of the violin is usually plotted resting on the horizontal axis. Here I want to emphasize the summary offered by the box plot, which shows in a line a location of the distribution (mean or median) and the scale of its spread (quartiles or standard deviation).

1-dimensional (single variable) Gaussian distribution

$$x \sim \mathcal{N}\left(\mu, \sigma^2\right) \propto e^{\frac{-(x-\mu)^2}{2\sigma^2}} \quad \mu, \sigma \in \mathbb{R}$$
$$\mu \equiv E[x] \quad \sigma^2 \equiv E[(x - E[x])(x - E[x])]$$



The expectation operation is linear, if a and b are constants:

$$E[a + bx] = a + bE[x]$$

Numerically, the expectation is approximated by the usual arithmetic mean

$$E[x] \simeq \frac{1}{N} \sum_{i=1}^{N} x_i$$

where x_i are the realizations (samples) of the random variable x. For random vectors, the sum is applied to each component.

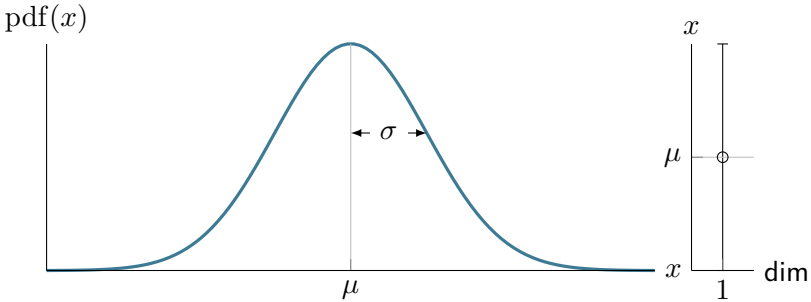
Check that the approximation is indeed linear!

The deviation from the mean (without compensation), i. e. variance, can be written as:

$$E[(x - E[x])(x - E[x])] = E[x^2 - 2xE[x] + E[x]^2] =$$
$$E[x^2] - 2E[x]^2 + E[x]^2 = E[x^2] - E[x]^2$$

1-dimensional (single variable) Gaussian distribution

$$x \sim \mathcal{N}\left(\mu, \sigma^2\right) \propto e^{\frac{-(x-\mu)^2}{2\sigma^2}} \quad \mu, \sigma \in \mathbb{R}$$
$$\mu \equiv E[x] \quad \sigma^2 \equiv E[(x - E[x])(x - E[x])]$$



The left panel shows the usual way of representing the body of the violin we saw before. The right panel shows the box plot view, in this case showing the mean value and the standard deviation.

The curve can indicate the frequency at which values would appear if we take large (infinite) number of samples from the distribution. Samples appearing more frequently in regions with higher values. It can also be used to represent our knowledge about a magnitude without the need to make a reference to sampling. The former is the frequentist interpretation, the latter is aligned with the Bayesian view.

1-dimensional (single variable) Gaussian distribution

Linearly* transformed (or linear change of) variable

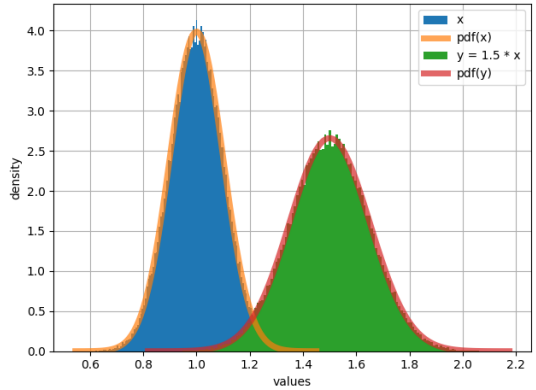
With a linear change of variables, it maps to another gaussian:

$$x \sim \mathcal{N}\left(\mu, \sigma^2\right)$$
$$y = ax + b$$
$$y \sim \mathcal{N}\left(a\mu + b, a^2\sigma^2\right)$$

$$\frac{-(x-\mu)^2}{2\sigma^2} = \frac{-(\frac{y-b}{a} - \mu)^2}{2\sigma^2} = \frac{-(y-b-a\mu)^2}{2a^2\sigma^2}$$

$$\mu_y := a\mu + b, \quad \sigma_y^2 := a^2\sigma^2$$

μ transforms like the variable and σ^2 is multiplied by the squared slope



source: *s_sampling_mapping.m*

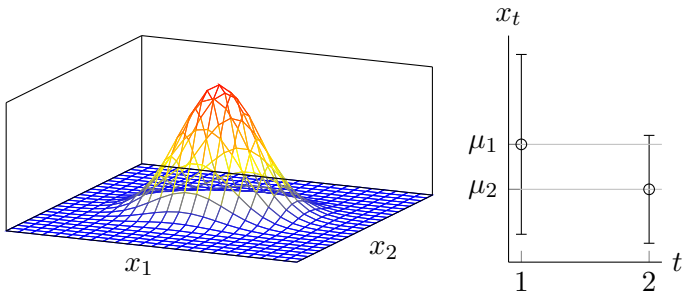
Also via the expectation operator:

$$\begin{aligned}\sigma_y^2 =& E[y^2] - E[y]^2 = E[(ax+b)^2] - E[ax+b]^2 \\ =& E[a^2x^2 + 2abx + b^2] - (a^2E[x]^2 + 2abE[x] + b^2) \\ =& a^2E[x^2] + 2abE[x] + b^2 - a^2E[x]^2 - 2abE[x] - b^2 \\ =& a^2(E[x^2] - E[x]^2) = a^2\sigma_x^2\end{aligned}$$

2-dimensional (two variables) Gaussian distribution

$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \propto \exp\left[\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right]$$

$\boldsymbol{\mu} \in \mathbb{R}^{2\times 1},\ \boldsymbol{\Sigma} \in \mathbb{R}^{2\times 2}$



See `s_mvnormal.py`

We can still plot the joint distribution of the two variables on a piece of paper. However it can be quite difficult to read.
The box plot or violin view is also useful. We loose the information about the interaction between the variables.
The covariance matrix has the entries:

$$\boldsymbol{\Sigma} = \begin{bmatrix} \mathrm{var}(x_1) & \mathrm{var}(x_1, x_2) \\ \mathrm{var}(x_1, x_2) & \mathrm{var}(x_2) \end{bmatrix}$$

2-dimensional (two variables) Gaussian distribution

Linearly* transformed (or linear change of) variable

$$\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$
$$\boldsymbol{y} = A\boldsymbol{x} + \boldsymbol{b}$$
$$\boldsymbol{y} \sim \mathcal{N}\left(A\boldsymbol{\mu} + \boldsymbol{b}, A\boldsymbol{\Sigma}A^\top\right)$$

$$\operatorname{var}(\boldsymbol{y}) \coloneqq E[(\boldsymbol{y} - E[\boldsymbol{y}])\,(\boldsymbol{y} - E[\boldsymbol{y}])^\top]$$

$$\begin{aligned}\operatorname{var}(A\boldsymbol{x} + \boldsymbol{b}) \coloneqq & E[(A\boldsymbol{x} + \boldsymbol{b} - E[A\boldsymbol{x} + \boldsymbol{b}])\,(A\boldsymbol{x} + \boldsymbol{b} - E[A\boldsymbol{x} + \boldsymbol{b}])^\top] \\ = & E[(A\boldsymbol{x} - E[A\boldsymbol{x}])\,(A\boldsymbol{x} - E[A\boldsymbol{x}])^\top] \\ = & A\,(E[\boldsymbol{x} - E[\boldsymbol{x}])\,(\boldsymbol{x} - E[\boldsymbol{x}])^\top]\,A^\top = A\operatorname{var}(\boldsymbol{x})A^\top\end{aligned}$$

Note: this is valid for any number of dimensions

The definition of variance follows the same idea as before. The factors $\boldsymbol{y} - E[\boldsymbol{y}]$ are the deviation from the mean of each component. The product by the transpose pairs all components to each other. The diagonal terms are

$$\boldsymbol{\Sigma}_{i,i} = E[(y_i - E[y_i])(y_i - E[y_i])]$$

called covaraice of y_i. The off-diagonal terms are

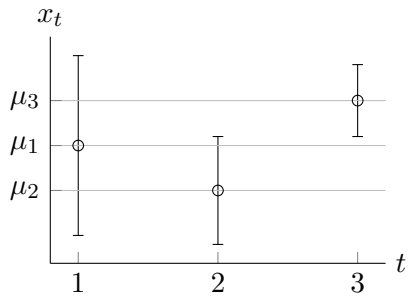
$$\boldsymbol{\Sigma}_{i,j} = E[(y_i - E[y_i])(y_j - E[y_j])]$$

called (cross-)covariances of y_i and y_j

3-dimensional Gaussian distribution

$$\boldsymbol{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \propto \exp\left[\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right]$$

$$\boldsymbol{\mu} \in \mathbb{R}^{3\times 1},\ \boldsymbol{\Sigma} \in \mathbb{R}^{3\times 3}$$



With three variables we cannot plot the joint distribution on a piece of paper. The box plot or violin view is still useful.

multi-dimensional distribution

Box or violin plot view

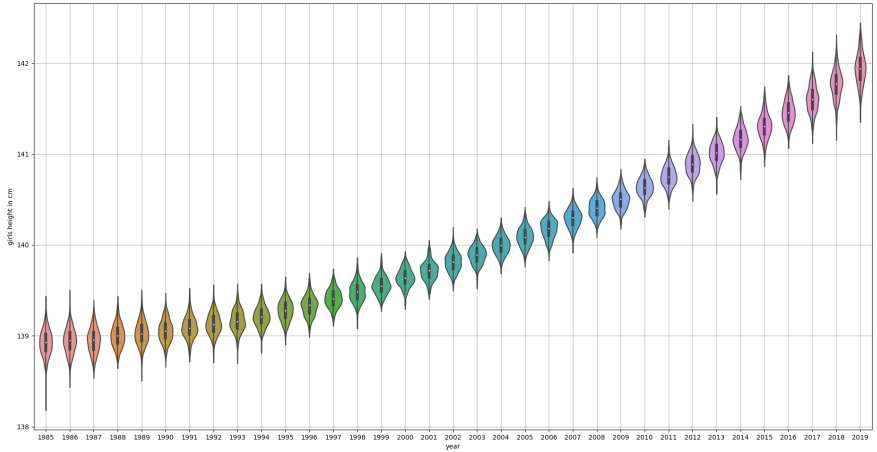
year	height
1985	138.8
1986	139.0
1987	139.0
1988	138.8
⋮	⋮
2010	138.8
2011	138.7
2012	139.2
2013	139.0
2014	139.0
2015	138.6
⋮	⋮

Recall the height data, it was also indicated the year in which the measurement was done. Hence we can think of the heights of a given year as a random variable. We will have as many variables as years in our data set.

We cannot plot the joint distribution of all these variables.

multi-dimensional distribution

Box or violin plot view

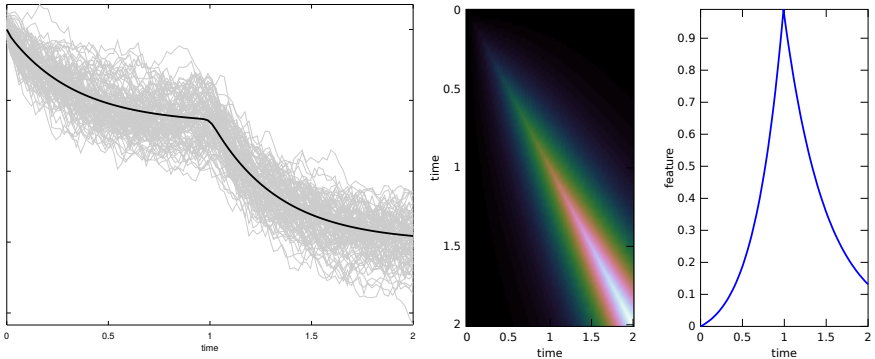


Kalman('60): "Intuitively, a random process is simply a set of random variables which are indexed in such a way as to bring the notion of time into the picture."

The violin plot of each year put in temporal order is used to show the evolution of some statistical properties (summaries) of the height. The visualization hints at a single variable (height) that changes over time. to understand the formal treatment, however, we need to think as a collection of indexed random variables (one for each year). The two ways of looking at the plot are compatible, but questions like: "What's the correlation between the height in 1995 and 2006?" are better understood when we think of different random variables.

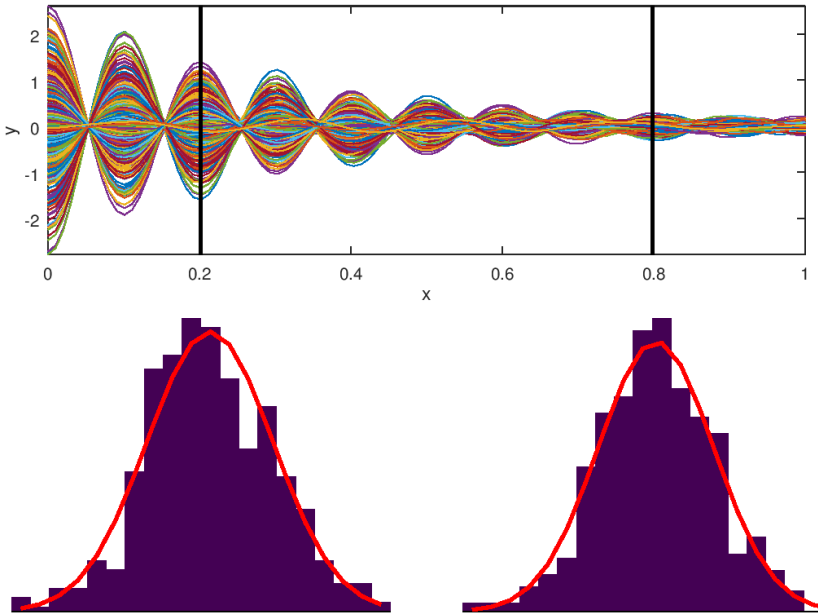
∞-dimensional: GP

$$\boldsymbol{\mu} \to m(t) \quad \boldsymbol{\Sigma} \to k(t, t')$$
$$x(t) \sim \mathcal{GP}(m(t), k(t, t'))$$



A Gaussian process (or stochastic process if the local distribution is not specified) is the idea saw before taken to the limit in which the time index is continuous. The mean value becomes a function of time, and the covariance is a function of two times.

∞-dimensional: GP



Any slice in time of the process reveals a Gaussian distribution. These processes are generated by linear dynamical systems with Gaussian process noise.

Lunch: 1 hour

Iterated maps revisited

We combine what we have learned so far to define the prediction step of the Kalman filter. We also discuss summary statistics from samples of a stochastic process, and the exact computation of mean and variance for Gaussian processes.

Iterated map with process noise: single variable

Map (mapping, transformation, etc.):

$$x = \alpha y + \epsilon \quad \epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2)$$

What do we get if $\alpha = 1$?

What's the distribution of x if we know the value of y?

Iterated map with process noise: single variable

Map (mapping, transformation, etc.):

$$x = \alpha y + \epsilon \quad \epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2)$$

Iterated map (use the result as the next value of x):

$$x_k = \alpha x_{k-1} + \epsilon_k \quad |\alpha| \leq 1,\ \epsilon_k \sim \mathcal{N}(0, \sigma_\epsilon^2)$$

That we draw ϵ_k from a normal distribution means that each sample is independent. If we look at the collection of ϵ_k as a random vector, this means that the covariance matrix is diagonal:

$$\Sigma_{\boldsymbol{\epsilon}}(i,j) = \mathrm{cov}\left(\epsilon_i, \epsilon_j\right) = \sigma_\epsilon^2 \delta_{i,j}$$

The samples at different steps are independent and therefore the (cross-)correlation between samples at different steps is zero.

"The noise" $\boldsymbol{\epsilon}$ is called *process* noise, because it feeds into the evolution of the process. It models uncertainties and/or perturbations in the dynamics.

Iterated map with process noise: single variable

Map (mapping, transformation, etc.):

$$x = \alpha y + \epsilon \quad \epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2)$$

Iterated map (use the result as the next value of x):

$$x_k = \alpha x_{k-1} + \epsilon_k \quad |\alpha| \leq 1,\ \epsilon_k \sim \mathcal{N}(0, \sigma_\epsilon^2)$$

0. x_0

Iterated map with process noise: single variable

Map (mapping, transformation, etc.):

$$x = \alpha y + \epsilon \quad \epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2)$$

Iterated map (use the result as the next value of x):

$$x_k = \alpha x_{k-1} + \epsilon_k \quad |\alpha| \leq 1,\ \epsilon_k \sim \mathcal{N}(0, \sigma_\epsilon^2)$$

0. x_0
1. $x_1 = \alpha x_0 + \epsilon_1$

Iterated map with process noise: single variable

Map (mapping, transformation, etc.):

$$x = \alpha y + \epsilon \quad \epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2)$$

Iterated map (use the result as the next value of x):

$$x_k = \alpha x_{k-1} + \epsilon_k \quad |\alpha| \leq 1,\ \epsilon_k \sim \mathcal{N}(0, \sigma_\epsilon^2)$$

0. x_0
1. $x_1 = \alpha x_0 + \epsilon_1$
2. $x_2 = \alpha x_1 + \epsilon_2 = \alpha^2 x_0 + \alpha\epsilon_1 + \epsilon_2$

Iterated map with process noise: single variable

Map (mapping, transformation, etc.):

$$x = \alpha y + \epsilon \quad \epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2)$$

Iterated map (use the result as the next value of x):

$$x_k = \alpha x_{k-1} + \epsilon_k \quad |\alpha| \leq 1,\ \epsilon_k \sim \mathcal{N}(0, \sigma_\epsilon^2)$$

0. x_0
1. $x_1 = \alpha x_0 + \epsilon_1$
2. $x_2 = \alpha x_1 + \epsilon_2 = \alpha^2 x_0 + \alpha\epsilon_1 + \epsilon_2$
3. $x_3 = \alpha x_2 + \epsilon_3 = \alpha^3 x_0 + \alpha^2\epsilon_1 + \alpha\epsilon_2 + \epsilon_3$

Iterated map with process noise: single variable

Map (mapping, transformation, etc.):

$$x = \alpha y + \epsilon \quad \epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2)$$

Iterated map (use the result as the next value of x):

$$x_k = \alpha x_{k-1} + \epsilon_k \quad |\alpha| \leq 1,\ \epsilon_k \sim \mathcal{N}(0, \sigma_\epsilon^2)$$

0. x_0
1. $x_1 = \alpha x_0 + \epsilon_1$
2. $x_2 = \alpha x_1 + \epsilon_2 = \alpha^2 x_0 + \alpha\epsilon_1 + \epsilon_2$
3. $x_3 = \alpha x_2 + \epsilon_3 = \alpha^3 x_0 + \alpha^2\epsilon_1 + \alpha\epsilon_2 + \epsilon_3$
4. $\ldots$

Iterated map with process noise: single variable

Map (mapping, transformation, etc.):

$$x = \alpha y + \epsilon \quad \epsilon \sim \mathcal{N}(0, \sigma_\epsilon^2)$$

Iterated map (use the result as the next value of x):

$$x_k = \alpha x_{k-1} + \epsilon_k \quad |\alpha| \leq 1,\ \epsilon_k \sim \mathcal{N}(0, \sigma_\epsilon^2)$$

0. x_0
1. $x_1 = \alpha x_0 + \epsilon_1$
2. $x_2 = \alpha x_1 + \epsilon_2 = \alpha^2 x_0 + \alpha\epsilon_1 + \epsilon_2$
3. $x_3 = \alpha x_2 + \epsilon_3 = \alpha^3 x_0 + \alpha^2\epsilon_1 + \alpha\epsilon_2 + \epsilon_3$
4. $\ldots$
5. $x_k = \alpha^k x_0 + \sum_{i=1}^{k} \alpha^{k-i}\epsilon_i$

What happens to the "old" noise terms if $\alpha < 1$? And if $\alpha > 1$?

Take $\alpha = 10^{-1}$, $x_0 = 1$, and $\sigma_\epsilon = 10^{-2}$. We ask for what k is "likely" that $|\epsilon_k| > \alpha^k x_0$.
Taking "likely" as $P(|\epsilon_k| > \alpha^k x_0) > 0.9$, that is $P(|\epsilon_k| > 10^{-k})$, and using the cumulative distribution of the normal we get $k = 3$. After 3 steps the signal is just noise. The step correlation of the signal, however, still carries information about the map.

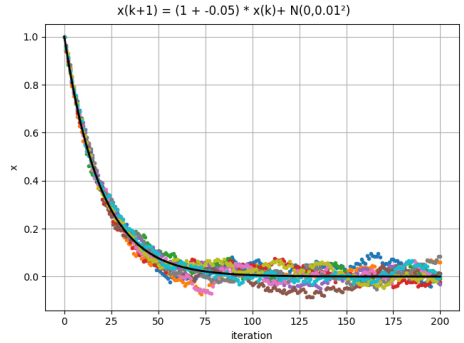
Iterated map with process noise: single variable

Example

$$x_k = (1 - 0.05)x_{k-1} + \epsilon_k \quad \epsilon_k \sim \mathcal{N}(0, 0.01^2)$$

- Different trajectories for the same initial value
- What's the formula for the mean trajectory (black line)?
- What's the formula for the variance of the trajectories?

source: *s_iterated_map1D.py*



The trajectories do not repeat, even for the same initial value. Each simulation is "unique". Running the map several times from the same initial value shows a distribution of trajectories (also called paths). We can compute the mean trajectory by averaging the ensemble of trajectories at each step. We can also compute the variance of the trajectories at each step.
Can we find a map for the mean trajectory?
Can we find a map for the variance?

Note the difference: we do not ask for the evolution of each initial value (a path) but for the evolution of the ensemble (not over time, for each time step!) mean and variance of many paths; in general the evolution of their distribution.

The result is is know as the Fokker-Planck equation (a.k.a. Kolmogorov forward equation, a.k.a. Smoluchowski equation).

Iterated map with process noise: single variable

Mean and Variance

$$x_k = \alpha^k x_0 + \sum_{i=1}^{k} \alpha^{k-i} \epsilon_i \quad \epsilon_i \sim \mathcal{N}(0, \sigma_\epsilon^2)$$

At each time step it is a linear mapping, the result is Gaussian.

$$\begin{aligned} E[x_k] =& E[\alpha^k x_0 + \sum_{i=1}^{k} \alpha^{k-i} \epsilon_i] = \alpha^k E[x_0] + \sum_{i=1}^{k} \alpha^{k-i} \underbrace{E[\epsilon_i]}_{=0} \\ =& \alpha^k E[x_0] \end{aligned}$$

The mean trajectory is the one of the map without noise

We directly compute the mean using the expectation operator.
Re-do the calculation using the iteration formula:

$$x_k = \alpha x_{k-1} + \epsilon_{k-1}$$

What is the iteration formula for the mean?

Iterated map with process noise: single variable

Mean and Variance

$$x_k = \alpha^k x_0 + \sum_{i=1}^{k} \alpha^{k-i} \epsilon_i \quad \epsilon_i \sim \mathcal{N}(0, \sigma_\epsilon^2)$$

At each time step it is a linear mapping, the result is Gaussian.

$$r_k \coloneqq x_k - E[x_k] = \alpha^k (x_0 - E[x_0]) + \overbrace{\sum_{i=1}^{k} \alpha^{k-i} \epsilon_i}^{\phi_k}$$

$$\begin{aligned} E[r_k^2] =& E[\alpha^{2k} (x_0 - E[x_0])^2 + 2\alpha^k (x_0 - E[x_0]) \phi_k + \phi_k^2] \\ =& \alpha^{2k} \underbrace{E[(x_0 - E[x_0])^2]}_{\sigma_0^2} + 2\alpha^k \underbrace{E[(x_0 - E[x_0]) \phi_k]}_{=0} + E[\phi_k^2] \\ =& \alpha^{2k} \sigma_0^2 + E\left[\left(\sum_{i=1}^{k} \alpha^{k-i} \epsilon_i\right)^2\right] \end{aligned}$$

To compute the variance we first define the residual r_k at each step. Then apply the expectation operator as usual.
The initial condition could have $\sigma_0 = 0$, I just kept it because it doesn't hurt.
In the last equality, all crossed terms are zero $E[\epsilon_i \epsilon_j] = \delta_{ij}$ (uncorrelated noise).

Iterated map with process noise: single variable

Mean and Variance

$$x_k = \alpha^k x_0 + \sum_{i=1}^{k} \alpha^{k-i} \epsilon_i \quad \epsilon_i \sim \mathcal{N}(0, \sigma_\epsilon^2)$$

At each time step it is a linear mapping, the result is Gaussian.

$$r_k \coloneqq x_k - E[x_k] = \alpha^k (x_0 - E[x_0]) + \overbrace{\sum_{i=1}^{k} \alpha^{k-i} \epsilon_i}^{\phi_k}$$

$$\begin{aligned} E[r_k^2] =& \alpha^{2k}\sigma_0^2 + E\left[\left(\sum_{i=1}^{k} \alpha^{k-i}\epsilon_i\right)^2\right] = \alpha^{2k}\sigma_0^2 + \sum_{i=1}^{k} \alpha^{2(k-i)} E[\epsilon_i^2] \\ =& \left(\alpha^2\right)^k \sigma_0^2 + \sum_{i=1}^{k} \left(\alpha^2\right)^{k-i} \sigma_\epsilon^2 \end{aligned}$$

Same as the state iteration but with α^2 and noise σ_ϵ^2.

Re-do the calculation using the iteration formula:

$$x_k = \alpha x_{k-1} + \epsilon_{k-1}$$

What is the iteration formula for the variance?

Iterated map: two variables

Map (mapping, transformation, etc.)

$$\boldsymbol{y} = \mathbf{A}\boldsymbol{x} + \boldsymbol{\epsilon} \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{\Sigma}_\epsilon)$$

Iterated map:

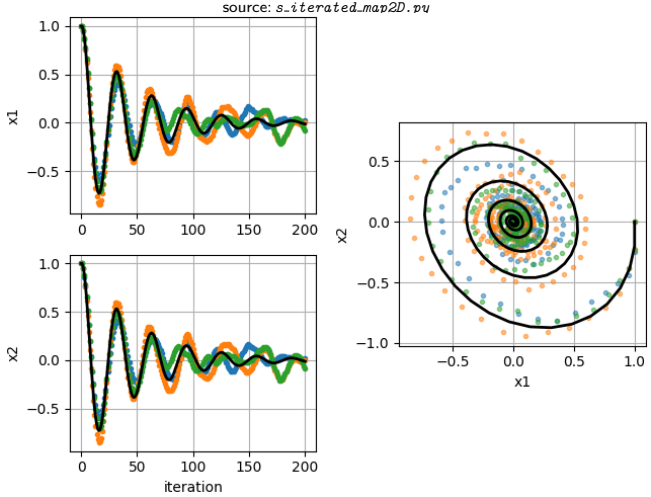
$$\boldsymbol{x}_k = \mathbf{A}\boldsymbol{x}_{k-1} + \boldsymbol{\epsilon}_k \quad \boldsymbol{\epsilon}_k \sim \mathcal{N}(0, \boldsymbol{\Sigma}_\epsilon)$$

Iterated map: two variables

Example

$$\boldsymbol{x}_k = \begin{bmatrix} 1 & 0.2 \\ -0.2 & 0.92 \end{bmatrix} \boldsymbol{x}_{k-1} + \boldsymbol{\epsilon}_k \quad \boldsymbol{\epsilon}_k \sim \mathcal{N}\left(0, \begin{bmatrix} \approx 0 & 0 \\ 0 & 0.03^2 \end{bmatrix}\right)$$

source: *s_iterated_map2D.py*



Iterated map: two variables

Mean and variance

$$\boldsymbol{x}_k = \mathbf{A}\boldsymbol{x}_{k-1} + \boldsymbol{\epsilon}_k \quad \boldsymbol{\epsilon}_k \sim \mathcal{N}(0, \boldsymbol{\Sigma}_\epsilon)$$

$$\boldsymbol{\mu}_k \coloneqq E[\boldsymbol{x}_k] = \mathbf{A}E[\boldsymbol{x}_{k-1}] = \mathbf{A}\boldsymbol{\mu}_{k-1}$$

$$\begin{aligned}
\boldsymbol{r}_k &\coloneqq \boldsymbol{x}_k - \boldsymbol{\mu}_k = \mathbf{A}\left(\boldsymbol{x}_{k-1} - \boldsymbol{\mu}_{k-1}\right) + \boldsymbol{\epsilon}_k = \mathbf{A}\boldsymbol{r}_{k-1} + \boldsymbol{\epsilon}_k \\
\boldsymbol{\Sigma}_k &\coloneqq E[\boldsymbol{r}_k \boldsymbol{r}_k^\top] = E[\left(\mathbf{A}\boldsymbol{r}_{k-1} + \boldsymbol{\epsilon}_k\right)\left(\mathbf{A}\boldsymbol{r}_{k-1} + \boldsymbol{\epsilon}_k\right)^\top] = \\
&= E[\mathbf{A}\boldsymbol{r}_{k-1}\boldsymbol{r}_{k-1}^\top\mathbf{A}^\top + \mathbf{A}\boldsymbol{r}_{k-1}\boldsymbol{\epsilon}_k^\top + \boldsymbol{\epsilon}_k\boldsymbol{r}_{k-1}^\top\mathbf{A}^\top + \boldsymbol{\epsilon}_k\boldsymbol{\epsilon}_k^\top] = \\
&= \mathbf{A}E[\boldsymbol{r}_{k-1}\boldsymbol{r}_{k-1}^\top]\mathbf{A}^\top + \mathbf{A}E[\boldsymbol{r}_{k-1}\boldsymbol{\epsilon}_k^\top] + E[\boldsymbol{\epsilon}_k\boldsymbol{r}_{k-1}^\top]\mathbf{A}^\top + E[\boldsymbol{\epsilon}_k\boldsymbol{\epsilon}_k^\top] = \\
&= \mathbf{A}\boldsymbol{\Sigma}_{k-1}\mathbf{A}^\top + \boldsymbol{\Sigma}_\epsilon
\end{aligned}$$

$$E[\boldsymbol{r}_{k-1}\boldsymbol{\epsilon}_k^\top] = E[(\boldsymbol{x}_k - \boldsymbol{\mu}_k)\boldsymbol{\epsilon}_k^\top] = E[\boldsymbol{x}_k\boldsymbol{\epsilon}_k^\top] - \boldsymbol{\mu}_k E[\boldsymbol{\epsilon}_k^\top] = 0$$

We directly compute the mean and variance by application of the expectation. To simplify the algebra in the variance we use the residuals $\boldsymbol{r}_k$. The cross-covaraince between residuals and noise at different steps $E[\boldsymbol{r}_{k-1}\boldsymbol{\epsilon}_k^\top]$ are zero because the noise has zero mean and is not correlated with the state.

Iterated map: important results

The iteration is a dynamical model of a process:

$$\boldsymbol{x}_k = \mathbf{A}\boldsymbol{x}_{k-1} + \boldsymbol{\epsilon}_k \quad \boldsymbol{\epsilon}_k \sim \mathcal{N}(0, \boldsymbol{\Sigma}_\epsilon)$$

Mean trajectory and variance:

$$\boldsymbol{\mu}_k = \mathbf{A}\boldsymbol{\mu}_{k-1} \qquad \boldsymbol{\Sigma}_k = \mathbf{A}\boldsymbol{\Sigma}_{k-1}\mathbf{A}^\top + \boldsymbol{\Sigma}_\epsilon$$

In the literature usually are written:

$$\boldsymbol{m}_k = \mathbf{A}\boldsymbol{m}_{k-1} \qquad \mathbf{P}_k = \mathbf{A}\mathbf{P}_{k-1}\mathbf{A}^\top + \mathbf{Q}$$

together with a measurement model:

$$\boldsymbol{y}_k = \mathbf{H}\boldsymbol{m}_k + \boldsymbol{\rho}_k \quad \boldsymbol{\rho}_k \sim \mathcal{N}(0, \mathbf{R})$$

are used to model a process and its sensors.

The notation used in many books on Bayesian filtering and smoothing is shown. The m used for the state reminds us that it is a "mean" state.

Note that the covariance of the noise can also be step dependent, as long as it does not depend on the state of the system.

So far we refer to $\boldsymbol{\epsilon}$ as "the noise", in the literature it is called *process* noise, because it feeds into the evolution of the process. It models uncertainties and/or perturbations in the dynamics.

Kalman filter: prediction step

The iteration is a dynamical model of the mean of a process and its sensors:

$$\boldsymbol{m}_k = \mathbf{A}\boldsymbol{m}_{k-1} \qquad \mathbf{P}_k = \mathbf{A}\mathbf{P}_{k-1}\mathbf{A}^\top + \mathbf{Q}$$
$$\bar{\boldsymbol{y}}_k = \mathbf{H}\boldsymbol{m}_k \qquad \mathbf{S}_k = \mathbf{H}\mathbf{P}_k\mathbf{H}^\top + \mathbf{R}$$

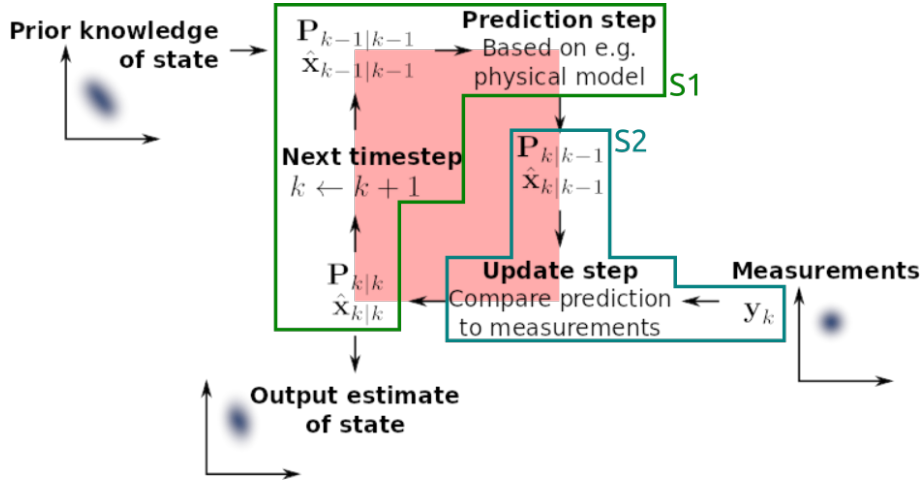
The first line is the **prediction step**. The second line provides a Gaussian measurement $\sim \mathcal{N}(\mathbf{H}\boldsymbol{m}_k, \mathbf{S}_k)$
To completely define the prediction step we need

- $\boldsymbol{m}$: state vector
- $\mathbf{P}$: state covariance
- $\mathbf{A}$: transition (system, dynamics) matrix $\rightarrow$evolves the state
- $\mathbf{Q}$: process noise covariance $\rightarrow$state noise inserted at each iteration
- $\mathbf{H}$: measurement matrix $\rightarrow$converts state to measurements
- $\mathbf{R}$: measurement noise covariance $\rightarrow$models sensor noise

All matrices can change in each iteration as long as they do not explicitly depend on the state $\boldsymbol{m}_k$.

1. What matrices are missing?
2. Derive the formulas for the mean measurement and the covariance of the measurements: $\boldsymbol{y}_k = \mathbf{H}\boldsymbol{m}_k \quad \mathbf{S}_k = \mathbf{H}\mathbf{P}_k\mathbf{H}^\top + \mathbf{R}$
3. What noises are included in $\mathbf{S}_k$?

Seminar overview



The first part of the seminar covered the modeling aspects that lead to the prediction step of the Kalman filter: how to advance the model starting from a "known" state. At this point you should be able to use the Kalman filter in applications and intuitively understand the effect of the different matrices in the setup.

Kalman filter: prediction step

Focus

This slide summarizes the modeling and prediction step of the KF.

The iteration represent a dynamical model of a process:

$$\boldsymbol{x}_k = \mathbf{A}\boldsymbol{x}_{k-1} + \boldsymbol{\epsilon}_k \quad \boldsymbol{\epsilon}_k \sim \mathcal{N}(0, \mathbf{Q}_k)$$
$$\boldsymbol{y}_k = \mathbf{H}\boldsymbol{x}_k + \boldsymbol{\rho}_k \quad \boldsymbol{\rho}_k \sim \mathcal{N}(0, \mathbf{R}_k)$$

Mean and variance of forward predictions (prediction step)

$$\boldsymbol{m}_k = \mathbf{A}\boldsymbol{m}_{k-1} \qquad \mathbf{P}_k = \mathbf{A}\mathbf{P}_{k-1}\mathbf{A}^\top + \mathbf{Q}_{k-1}$$

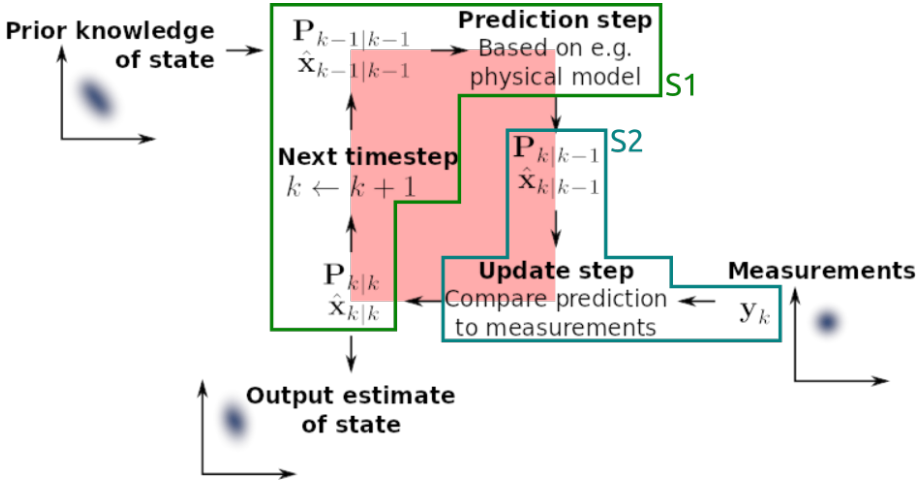
Coffee break: 15 minutes

Setting up the Kalman filter

Refer to the Jupyter notebooks in the repository. We will explore applications of the Kalman filter and learn how ot set it up. How it learns from the data will remain a mistery, but motors are a mistery to many drivers!

Wrap-up

Seminar overview



The first part of the seminar covered the modeling aspects that lead to the prediction step of the Kalman filter: how to advance the model starting from a "known" state. At this point you should be able to use the Kalman filter in applications and intuitively understand the effect of the different matrices in the setup.

Kalman filter: prediction step

The iteration is a dynamical model of the mean of a process and its sensors:

$$\boldsymbol{m}_k = \mathbf{A}\boldsymbol{m}_{k-1} \qquad \mathbf{P}_k = \mathbf{A}\mathbf{P}_{k-1}\mathbf{A}^\top + \mathbf{Q}$$
$$\bar{\boldsymbol{y}}_k = \mathbf{H}\boldsymbol{m}_k \qquad \mathbf{S}_k = \mathbf{H}\mathbf{P}_k\mathbf{H}^\top + \mathbf{R}$$

The first line is the **prediction step**. The second line provides a Gaussian measurement $\sim \mathcal{N}(\mathbf{H}\boldsymbol{m}_k, \mathbf{S}_k)$
To completely define the prediction step we need

- $\boldsymbol{m}$: state vector
- $\mathbf{P}$: state covariance
- $\mathbf{A}$: transition (system, dynamics) matrix $\rightarrow$evolves the state
- $\mathbf{Q}$: process noise covariance $\rightarrow$state noise inserted at each iteration
- $\mathbf{H}$: measurement matrix $\rightarrow$converts state to measurements
- $\mathbf{R}$: measurement noise covariance $\rightarrow$models sensor noise

All matrices can change in each iteration as long as they do not explicitly depend on the state $\boldsymbol{m}_k$.

1. What matrices are missing?
2. Derive the formulas for the mean measurement and the covariance of the measurements: $\boldsymbol{y}_k = \mathbf{H}\boldsymbol{m}_k \quad \mathbf{S}_k = \mathbf{H}\mathbf{P}_k\mathbf{H}^\top + \mathbf{R}$
3. What noises are included in $\mathbf{S}_k$?