Scalar Kalman Filter

This is an example of implementing a scalar version of the famous Kalman filter in Python.

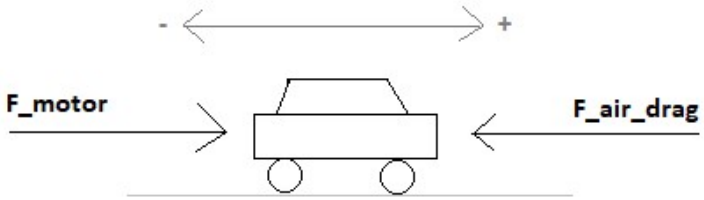
I highly recommend reading the Scalar Kalman filter article from Swarthmore if you don't have prior knowledge about the topic. It is most suitable for those who have knowledge about block diagrams and state-space representation of linear systems.

http://www.swarthmore.edu/NatSci/echeeve1/Ref/Kalman/ScalarKalman.html (http://www.swarthmore.edu/NatSci/echeeve1/Ref/Kalman/ScalarKalman.html)

The task in this example is to find an optimal estimate for the car speed at every time step using the speed mesurements and a mathematical model describing the change of the speed.

A simple car speed model

The basic Kalman filter requires that we have a mathematical model of the quantity we're estimating. The following free body diagram represents a car driving on a road. The force F_{motor} generated by the motor accelerates the car, and the air drag force F_{air_drag}, proportional to the velocity, slows the car. For the sake of simplicity all the other forces, such as frictions, are omitted. The objective is to represent the car speed as a differential equation.



Differential equation

The total force moving the car is obtained using the following equation, where a denotes the acceleration and m the mass of the car.

$$ma(t) = F_{motor}(t) - F_{air\_drag}(t)$$

We consider the air drag F_{air_drag} being proportinal to the velocity. β is a coefficient that depends on the properties of the air and the dimensions of the car.

$$F_{air\_drag}(t) = \beta v(t)$$
$$ma(t) = F_{motor}(t) - \beta v(t)$$

Since the acceleration is the derivative of the velocity, the equation can be rewriten as a differential equation. $\dot{v}$ denotes the time derivative of the velocity.

$$m\dot{v}(t) = F_{motor}(t) - \beta v(t)$$

Lets solve the equation for $\dot{v}$ to make the discretization process easier.

$$\dot{v}(t) = -\frac{\beta}{m}v(t) + \frac{1}{m}F_{motor}(t)$$

Discretization

Lets convert our differential equation to a difference equation by approximating the derivative of the velocity. T denotes the sampling period.

$$\frac{v(k+1)-v(k)}{T} \approx \dot{v}(t)$$
$$\frac{v(k+1)-v(k)}{T} = -\frac{\beta}{m}v(k) + \frac{1}{m}F_{motor}(k)$$
$$v(k+1) = v(k) - \frac{T\beta}{m}v(k) + \frac{T}{m}F_{motor}(k)$$
$$v(k+1) = (1 - \frac{T\beta}{m})v(k) + \frac{T}{m}F_{motor}(k)$$

State-space model

To make the equation (and code) a bit cleaner lets define some constants.

$$a = 1 - \frac{T\beta}{m}$$
$$b = \frac{T}{m}$$

Note that the variable a does not denote the acceleration anymore! By convention the symbol a is used in the state-space format naming, and since we will not need to denote acceleration using a specific symbol anymore, the symbol a will from now on be used to denote the constant describing the system. More information on state-space models can be found at https://en.wikipedia.org/wiki/State-space_representation (https://en.wikipedia.org/wiki/State-space_representation).

We can now rewrite the difference equation as follows.

$$v(k+1) = av(k) + bF_{motor}(k)$$

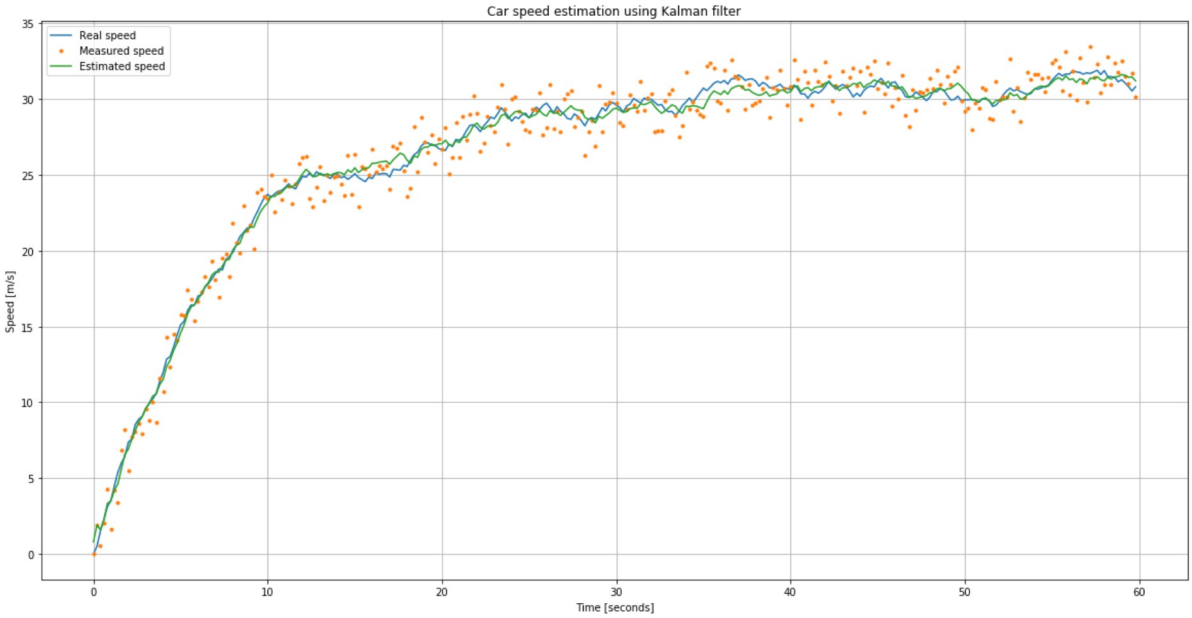
By convention the symbol x is used to represent the state variable and the symbol u the input variable in the state-space representation. Lets rename our variables and we're done! As a result we now have an equation which tells us what the speed of the car will be at the next time step if we know the current speed and the current input force.

$$x(k+1) = ax(k) + bu(k)$$

From here refere to the scalar Kalman filter proof at

http://www.swarthmore.edu/NatSci/echeeve1/Ref/Kalman/ScalarKalman.html (http://www.swarthmore.edu/NatSci/echeeve1/Ref/Kalman/ScalarKalman.html)

In [28]:
```
# Example of Kalman filtering for vehicle speed
# Joni Leppänen, 22.08.2019
#
# See the algorithm proof at:
# http://www.swarthmore.edu/NatSci/echeeve1/Ref/Kalman/ScalarKalman.html
#
# This script uses the same notation as in the website above except the
# Kalman gain is noted as 'G' instead of k, and the time index is noted
# as k instead of j. Also, do not confuse the sample period T in this code to
# the delay operator also noted as T in the block diagrams on the website.
import numpy as np
from matplotlib import pyplot as plt
import random

# Set seed for the random generator so that comparing the algorithm output when
# tweaking the parameters is easier.
random.seed(1)

# Simulation parameters
N = 300                         # Number of samples
T = 0.2                         # Sample period [seconds]
u = 6000                        # Gas (system input) [Newtons]

# Real vehicle parameters (the absolutely correct values for simulation)
m_real = 1500                   # Mass [kg]
beta_real = 200                 # Air drag
w_real = 0.4                    # Process noise standard deviation
r_real = 2                      # Measurement noise standard deviation

# Constants (for state space model)
a_real = 1 - T * beta_real / m_real  # System constant
b_real = T/m_real                    # Input constant

# Kalman filter parameters (must be measured or estimated somehow)
# Vehicle model constants.
m = 1500                        # Mass [kg]
beta = 200                      # Air drag
a = 1 - T * beta / m            # System constant
b = T/m                         # Input constant

# Noise parameters.
w = 0.4                         # Process noise standard deviation
r = 2                           # Measurement noise standard deviation
R = r**2                        # Measurement noise variance
Q = w**2                        # Process noise variance

bias = 0                        # Measurement bias
h = 1                           # Measurement gain

# Initial Kalman variance should be set to high so that the algorithm gives
# more weight to the measurements.
p = 1000                        # Initial Kalman variance

# Initial state (speed).
x_hat = 0

# Create the real vehicle speed vector and the measurement vector
v = np.zeros((N, 1))  # Real speed with process noise
z = np.zeros((N, 1))  # Speed measurement with measurement noise

for k in range(0, N - 1):
    v[k+1] = a_real*v[k] + b_real*u + w_real*(2*random.random()-1)
    z[k+1] = v[k+1] + r_real*(2*random.random()-1) + bias
```


Car speed estimation using Kalman filter
Real speed
Measured speed
Estimated speed
Speed [m/s]
Time [seconds]
0
5
10
15
20
25
30
35
0
10
20
30
40
50
60