Problem set 2

Denis Logashov

A. Before implementing anything, take a look at the code and answer the following theoretic and analytic questions (no coding). Write the value for the covariance Q of the noise added to the observation function, knowing that β is its standard deviation. To find out which is the value of β you should look at the default parameters passed to run.py lines 44 – 121. Write the equation for the covariance R_t of the noise added to the transition function, as explained in class and their corresponding numeric values for the initial robot command $u = [\delta_{rot1}, \delta_{trans}, \delta_{rot2}]^T = [0, 10, 0]^T$. Again, find out the initial values of α and their correction done in run.py line 152. Then derive the equations for the Jacobians G_t, V_t and H_t, and evaluate them at the initial mean state $\mu_1 = [x, y, \theta]^T = [180, 50, 0]^T$ as it is considered in run.py.

Solution

$Q = \beta^2 = deg2rad(20)^2 = 0.1218$

$$R_t = \begin{bmatrix} \alpha_1\delta_{rot1}^2 + \alpha_2\delta_{trans}^2 & 0 & 0 \\ 0 & \alpha_3\delta_{trans}^2 + \alpha_4(\delta_{rot1}^2 + \delta_{rot2}^2) & 0 \\ 0 & 0 & \alpha_1\delta_{rot2}^2 + \alpha_2\delta_{trans}^2 \end{bmatrix}$$

where $\alpha_1 = 0.05^2$, $\alpha_2 = 0.001^2$, $\alpha_3 = 0.05^2$, $\alpha_4 = 0.01^2$ and $u = [\delta_{rot1}, \delta_{trans}, \delta_{rot2}]^T = [0, 10, 0]^T$, then:

$$R_t = \begin{bmatrix} 0.05^2 \cdot 0^2 + 0.001^2 \cdot 10^2 & 0 & 0 \\ 0 & 0.05^2 \cdot 10^2 + 0.01^2(0^2 + 0^2) & 0 \\ 0 & 0 & 0.05^2 \cdot 0^2 + 0.001^2 \cdot 10^2 \end{bmatrix} = \begin{bmatrix} 0.0001 & 0 & 0 \\ 0 & 0.25 & 0 \\ 0 & 0 & 0.0001 \end{bmatrix}$$

To derive Jacobians, lets define transition and observation functions:

$$x_t = g(x_{t-1}, u_t) = \begin{bmatrix} x_{t-1} + \delta_{trans}\cos(\theta_{t-1} + \delta_{rot1}) \\ y_{t-1} + \delta_{trans}\sin(\theta_{t-1} + \delta_{rot1}) \\ \theta_{t-1} + \delta_{rot1} + \delta_{rot2} \end{bmatrix}$$

$$z_t = h(x_t) = \arctan 2(m_{j,y} - \bar{\mu}_{t,y}, m_{j,x} - \bar{\mu}_{t,x}) - \bar{\mu}_{t,\theta}$$

And now we can derive G_t, V_t and H_t:

$$G_t = \frac{\partial g(x_{t-1}, u_t)}{\partial x_{t-1}}\Big|_{\mu_{t-1}} = \begin{bmatrix} 1 & 0 & -\delta_{trans}\sin(\theta_{t-1} + \delta_{rot1}) \\ 0 & 1 & \delta_{trans}\cos(\theta_{t-1} + \delta_{rot1}) \\ 0 & 0 & 1 \end{bmatrix}$$

$$\partial/\partial x \qquad \partial/\partial y \qquad \partial/\partial\theta$$

$$V_t = \frac{\partial g(x_{t-1}, u_t)}{\partial u_t}\Big|_{\bar{u}_t} = \begin{bmatrix} -\delta_{trans}\sin(\theta_{t-1} + \delta_{rot1}) & \cos(\theta_{t-1} + \delta_{rot1}) & 0 \\ \delta_{trans}\cos(\theta_{t-1} + \delta_{rot1}) & \sin(\theta_{t-1} + \delta_{rot1}) & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

$$\partial/\partial\delta_{rot1} \qquad \partial/\partial\delta_{trans} \qquad \partial/\partial\delta_{rot2}$$

$$H_t = \left.\frac{\partial h(x_t)}{\partial x_t}\right|_{\bar{\mu}_t} = \begin{bmatrix} \frac{m_{j,y}-\bar{\mu}_{t,y}}{(m_{j,x}-\bar{\mu}_{t,x})^2+(m_{j,y}-\bar{\mu}_{t,y})^2} & -\frac{m_{j,x}-\bar{\mu}_{t,x}}{(m_{j,x}-\bar{\mu}_{t,x})^2+(m_{j,y}-\bar{\mu}_{t,y})^2} & -1 \end{bmatrix}$$
$$\partial/\partial\mu_x \qquad \partial/\partial\mu_y \qquad \partial/\partial\mu_\theta$$

where $u = [\delta_{rot1}, \delta_{trans}, \delta_{rot2}]^T = [0, 10, 0]^T$, $\mu_1 = [x, y, \theta]^T = [180, 50, 0]^T$ and $m_1 = [m_x, m_y]^T = [21, 0]^T$, then

$$G_1 = \begin{bmatrix} 1 & 0 & -10 \cdot \sin(0+0) \\ 0 & 1 & 10 \cdot \cos(0+0) \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 10 \\ 0 & 0 & 1 \end{bmatrix}$$

$$V_1 = \begin{bmatrix} -10 \cdot \sin(0+0) & \cos(0+0) & 0 \\ 10 \cdot \cos(0+0) & \sin(0+0) & 0 \\ 1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 10 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

As H_t is calculated using $\bar{\mu}$, then we must firstly calculate this $\bar{\mu}$:

$$\bar{\mu} = g(x_{t-1}, u_t) = \begin{bmatrix} x_0 + \delta_{trans}\cos(\theta_0 + \delta_{rot1}) \\ y_0 + \delta_{trans}\sin(\theta_0 + \delta_{rot1}) \\ \theta_0 + \delta_{rot1} + \delta_{rot2} \end{bmatrix} = \begin{bmatrix} 180 + 10 \cdot \cos(0+0) \\ 50 + 10 \cdot \sin(0+0) \\ 0+0+0 \end{bmatrix} = \begin{bmatrix} 190 \\ 50 \\ 0 \end{bmatrix}$$

And now we can calculate H_1:

$$H_1 = \begin{bmatrix} \frac{0-50}{(21-190)^2+(0-50)^2} & -\frac{21-190}{(21-190)^2+(0-50)^2} & -1 \end{bmatrix} = \begin{bmatrix} \frac{-50}{31061} & \frac{169}{31061} & -1 \end{bmatrix} = \begin{bmatrix} -0.0016 & 0.0054 & -1 \end{bmatrix}$$

C. Create plots of pose error versus time i.e., a plot of $\hat{x} - x$ vs. t, $\hat{y} - y$ vs. t, and $\hat{\theta} - \theta$ vs. t where $(\hat{x}, \hat{y}, \hat{\theta})$ is the filter estimated pose and $(x; y; \theta)$ is the ground-truth actual pose known only to the simulator. Plot the error in blue and in red plot the $\pm 3\sigma$ uncertainty bounds. Your state error should lie within these bounds approximately 99, 73% of the time (assuming Gaussian statistics). For the PF, use the sample mean and variance.

Solution

These plots and script for their generation are stored in the archive: plots_pf.png, plots_ekf.png and plots.py. The following plots shows the errors for EKF:

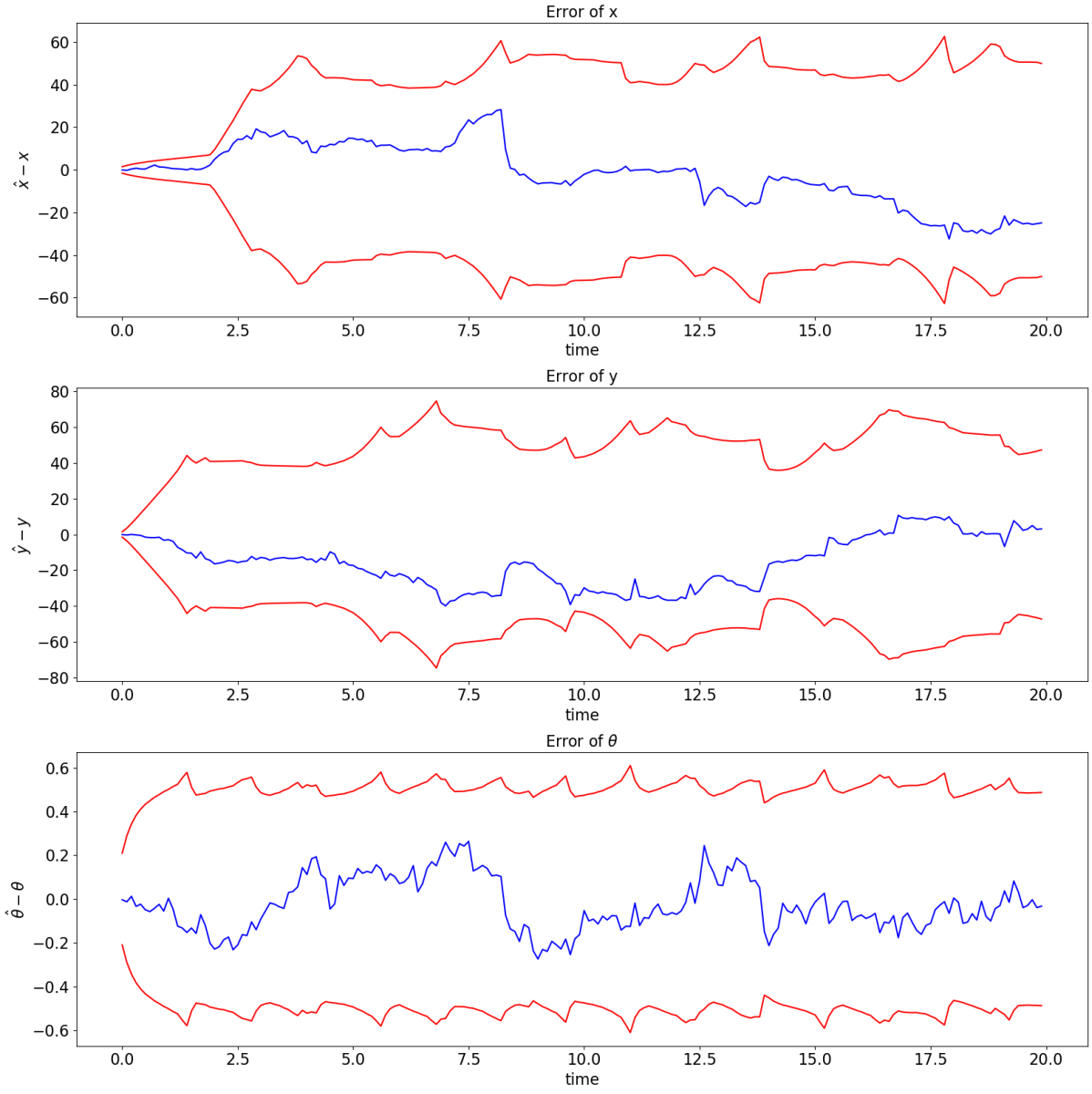

Figure 1: Errors for EKF

The Fig. 2 shows the errors for PF:

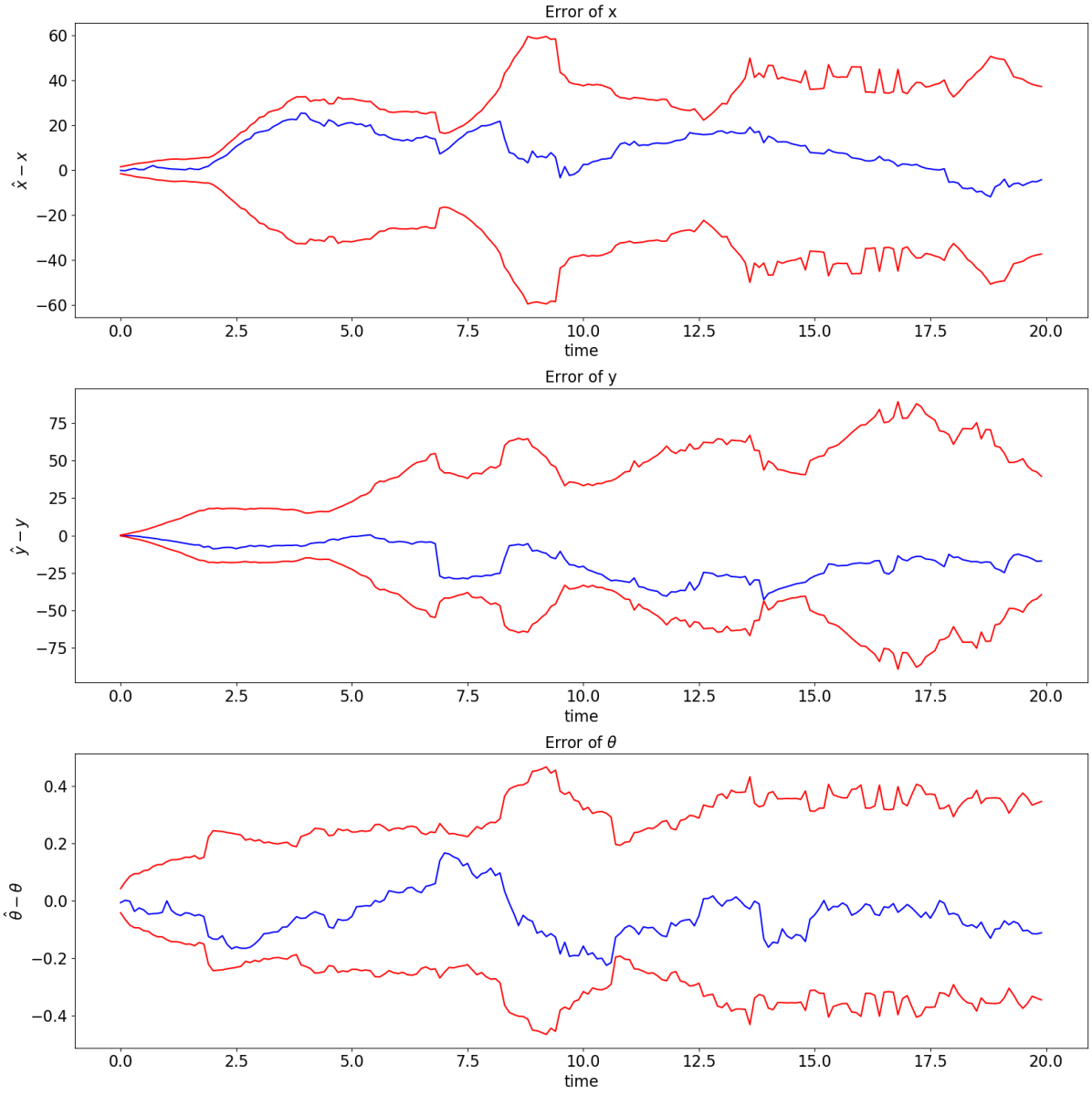

Figure 2: Errors for PF

D. Once your filters are implemented, please investigate some properties of them. How do they behave

- as the sensor or motion noise go toward zero?

  Let's look at the error of x with sensor noise go toward zero. As you can see on the Fig. 3 errors are almost equal.

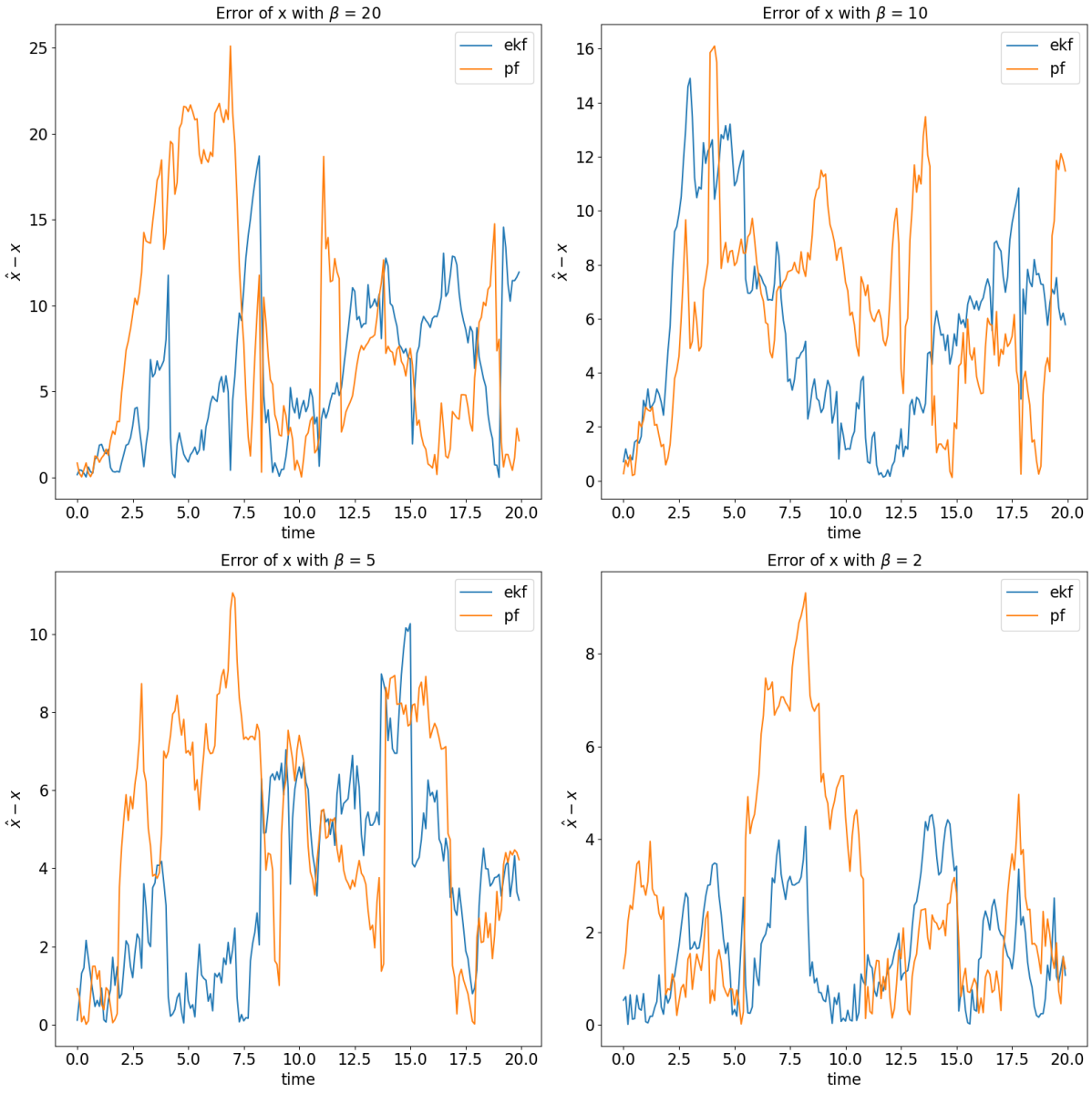

Figure 3: Errors of x

But if we will look at the covariance of measurements we can see, that covariance of PF affected more, i.e. less than covariance of EKF (Fig. 4, Fig. 5, Fig. 6).

From these graphs, we can see that the behavior of all states is almost the same, so in the future we will discuss the results using graphs of only one state. So, from the opposite side we can see on Fig. 7, that covariance of filters almost identical, with motion noise go toward to zero.

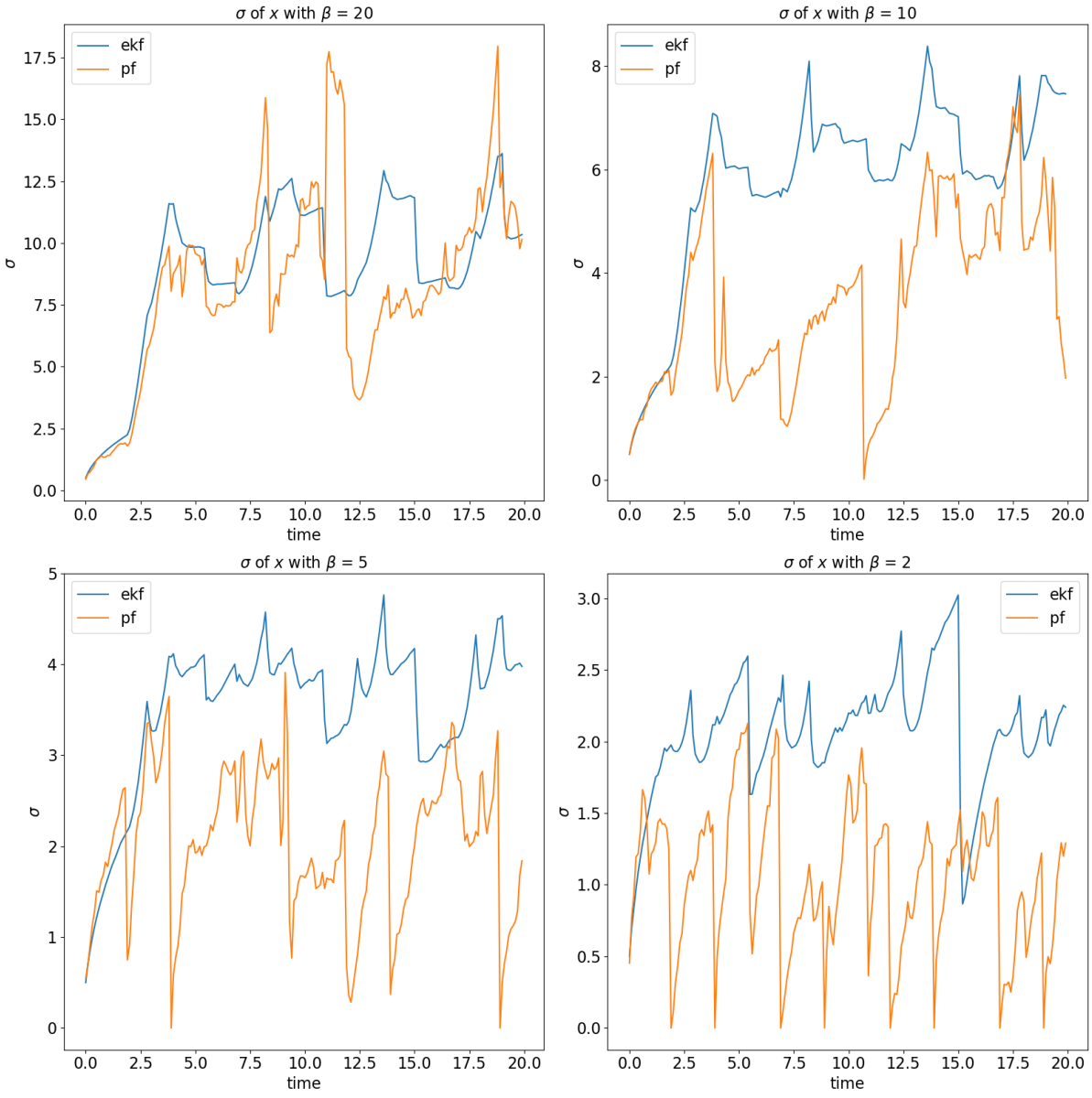

Figure 4: Standard deviation of x

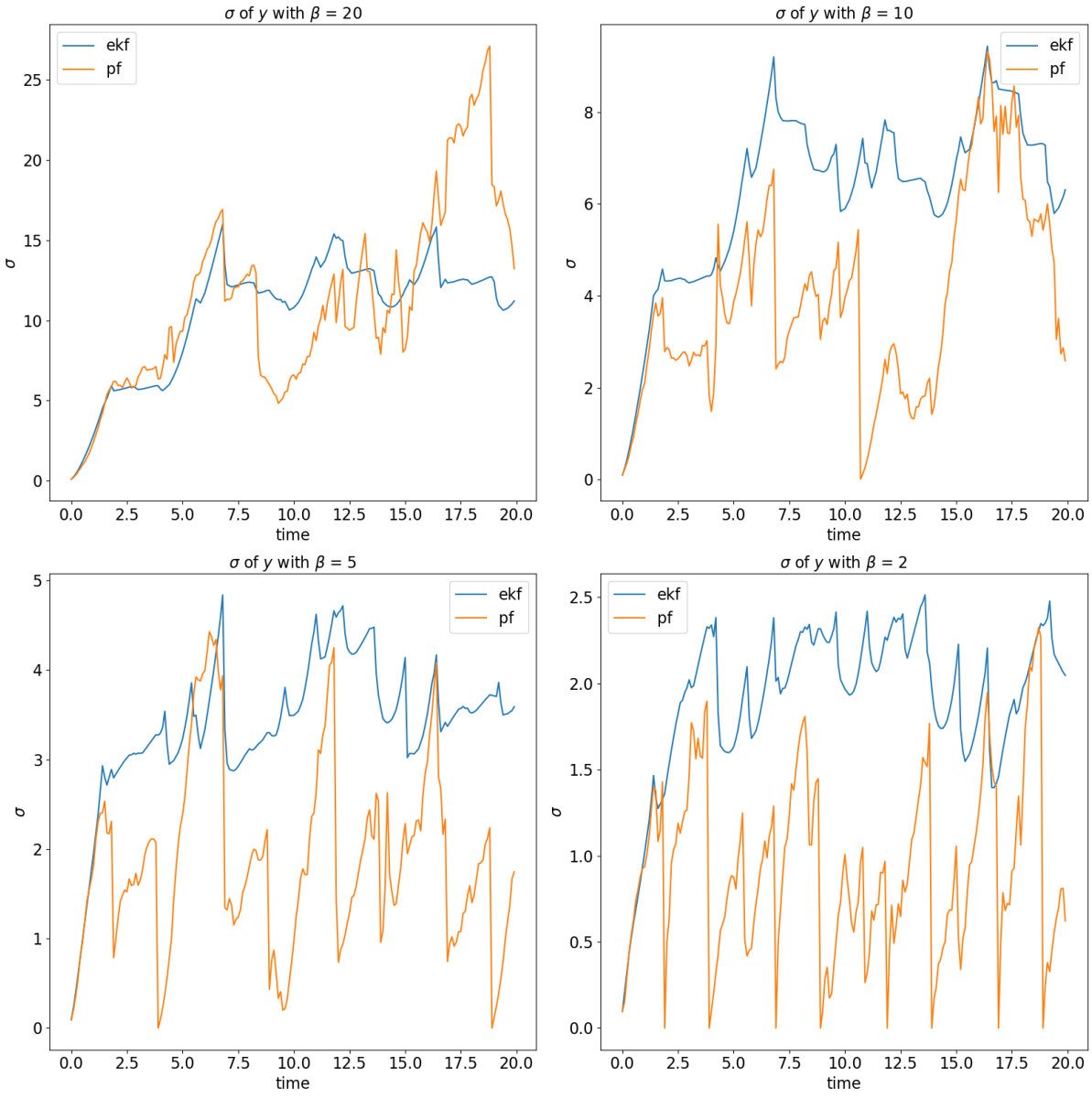

Figure 5: Standard deviation of y

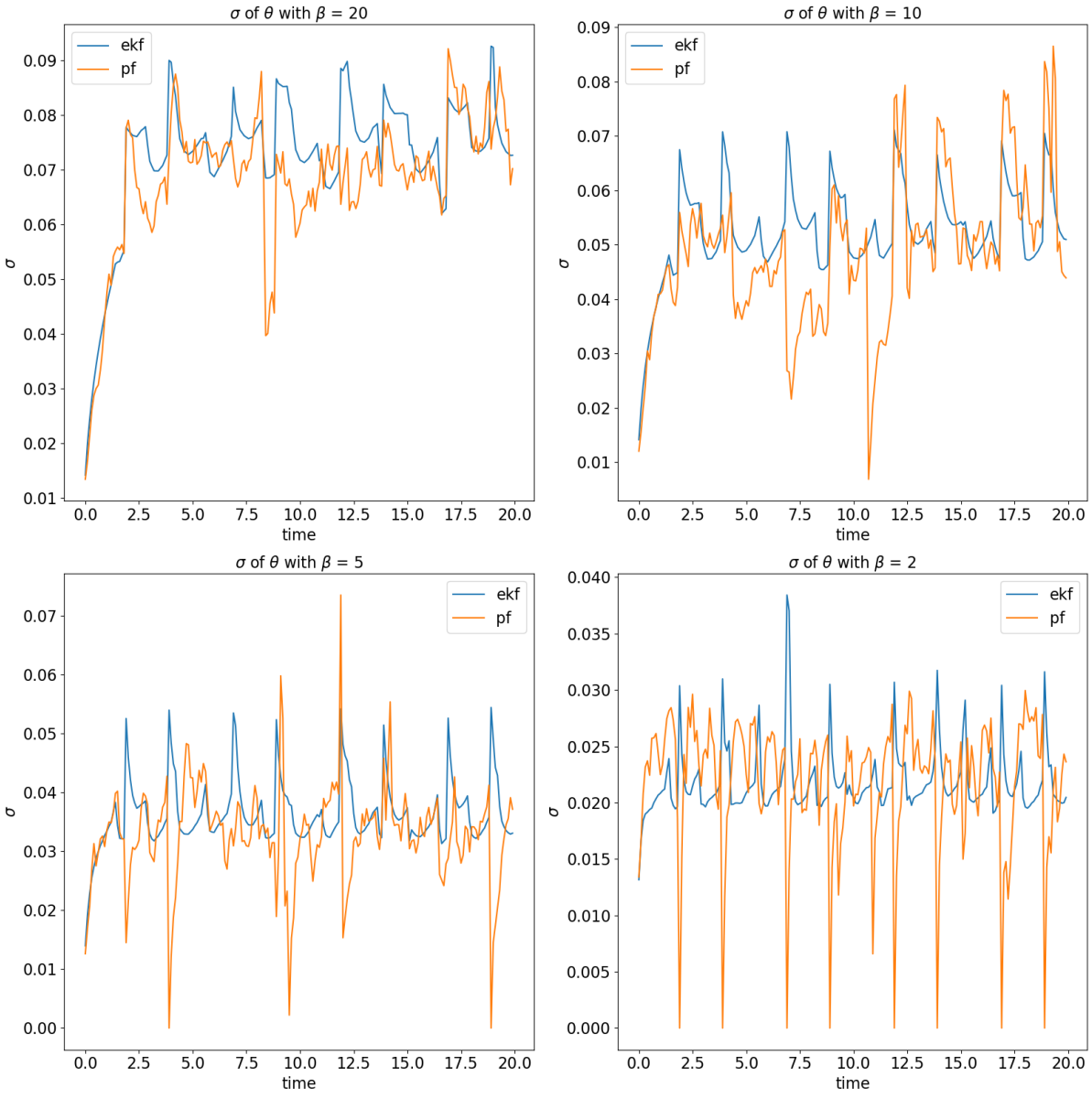

Figure 6: Standard deviation of θ

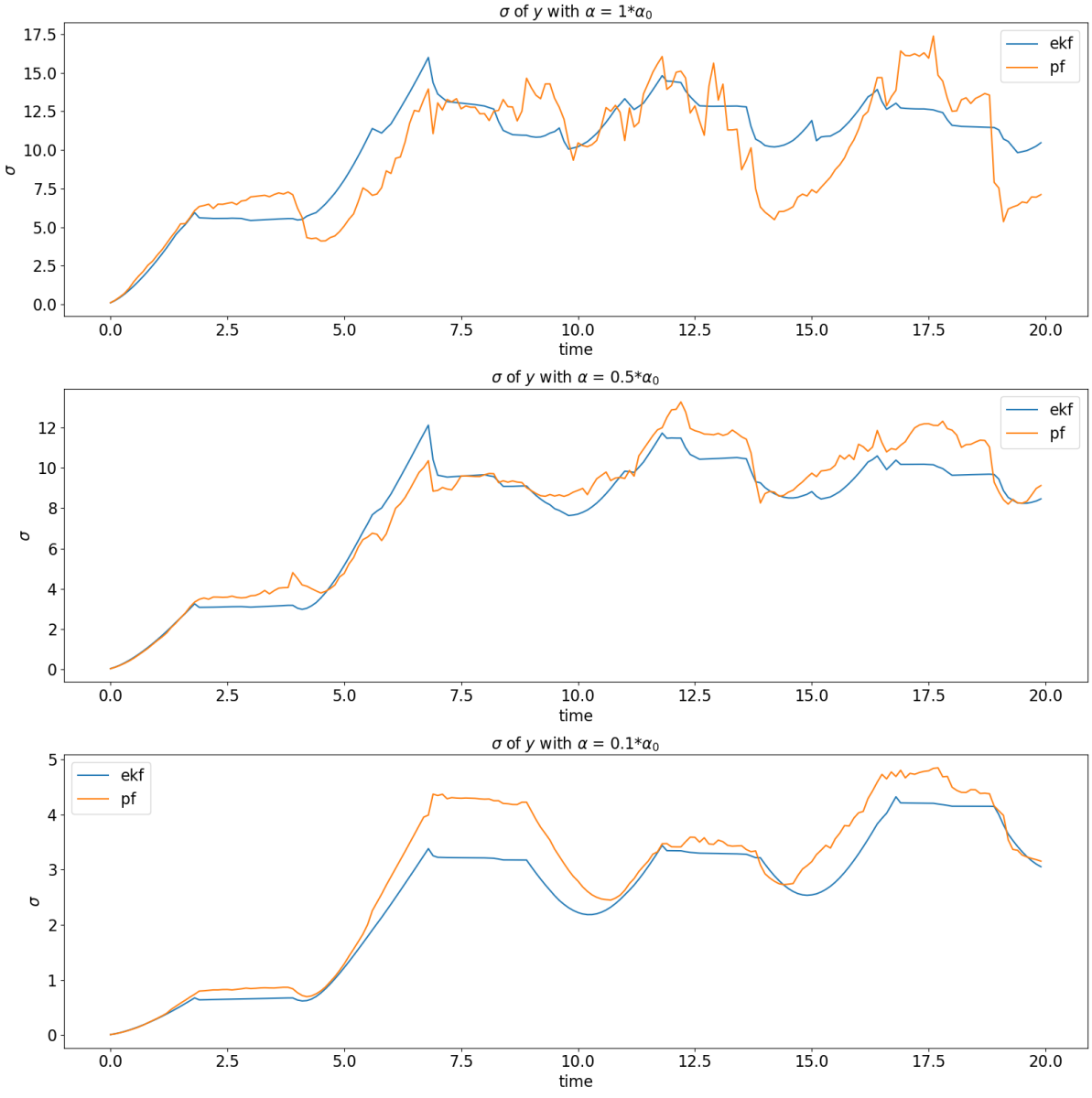

Figure 7: Standard deviation of y

But, as we can see on Fig. 8 errors of EKF dicreases faster than PF,

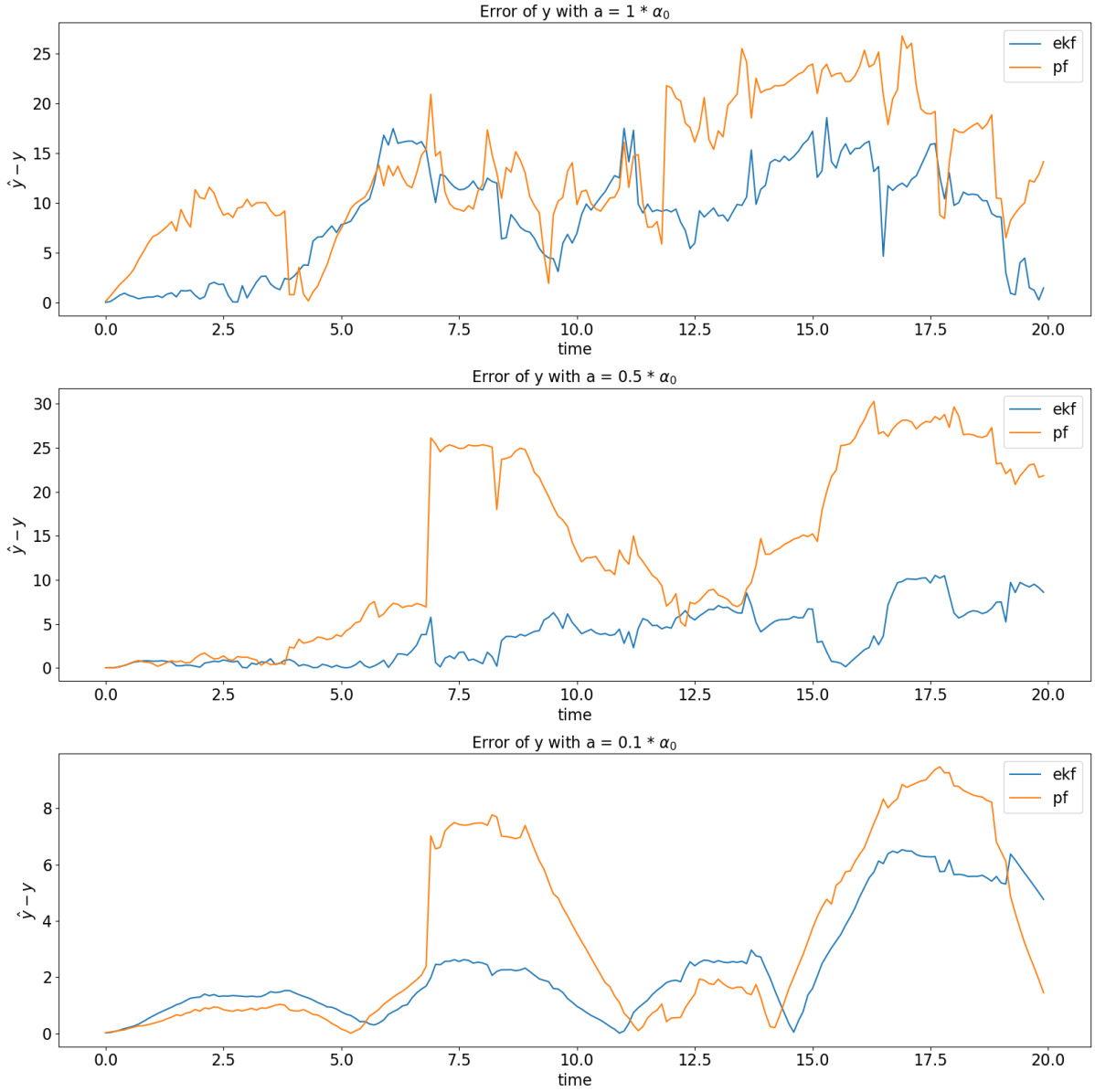

Figure 8: Errors of y

- as the number of particles decrease?

  With decreasing of number of particles PF become more unstable, since it has less different states for estimating, so errors increase (Fig. 9).

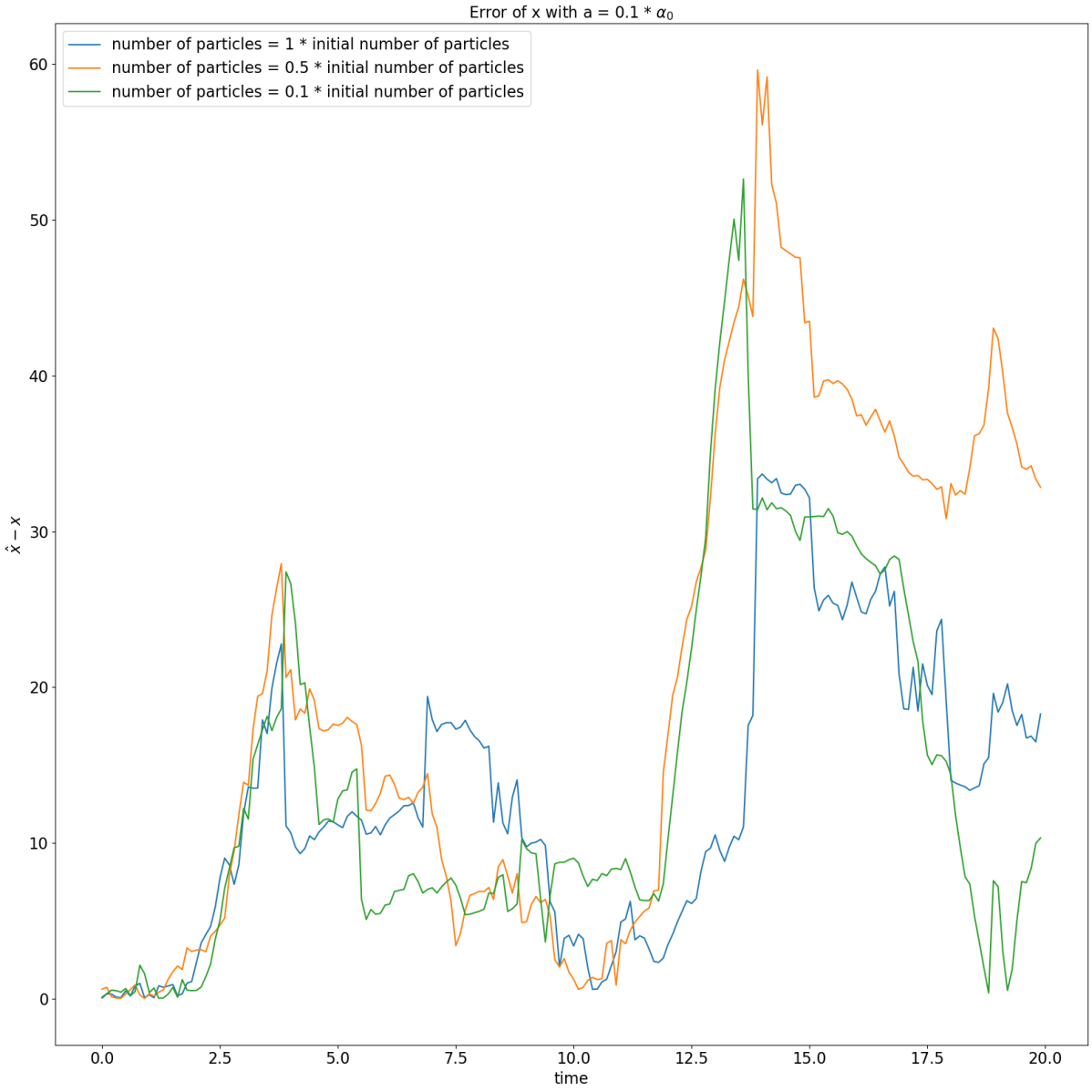

Figure 9: Errors of x

- if the filter noise parameters underestimate or overestimate the true noise parameters?

  If the sensor noise is underestimated for EKF, then we can see slightly grows of error but in the same time covariance decreases very fast, so filter becomes unstable (Fig. 10).

  At the same time with underestimating of noise for PF we can see grows of the error, since covariance go toward to zero (Fig. 11). So, PF afeected more by underestimating of sensor noise.

Figure 10: Errors and standard deviation of x

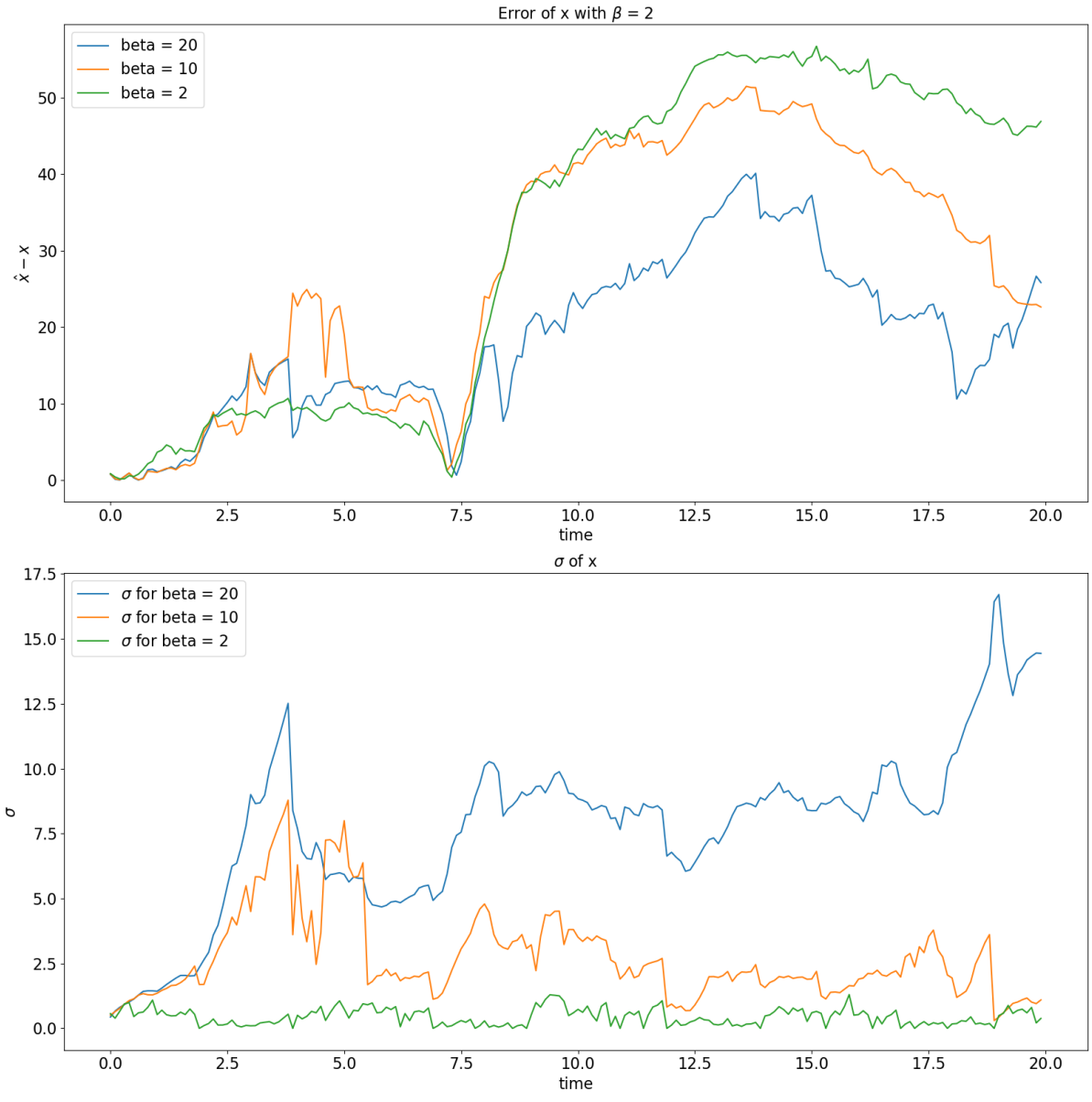

Figure 11: Errors and standard deviation of x

With an overestimated sensor noise, we can see an increase in covariance for both filters, but the effect on EKF is greater. The Fig. 12 shows changes for EKF and Fig. 13 shows changes for PF.

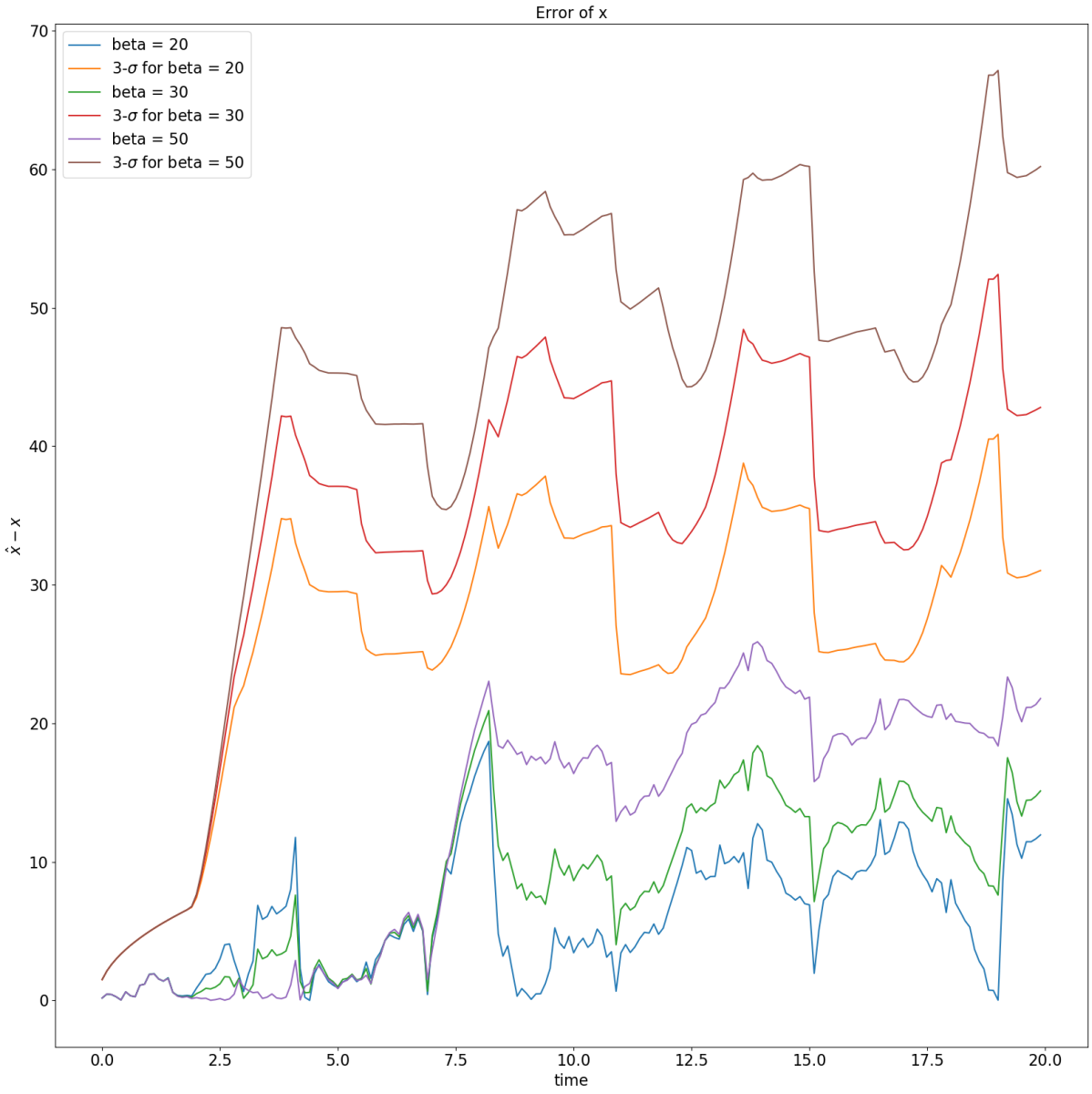

Figure 12: Errors and 3-σ of x

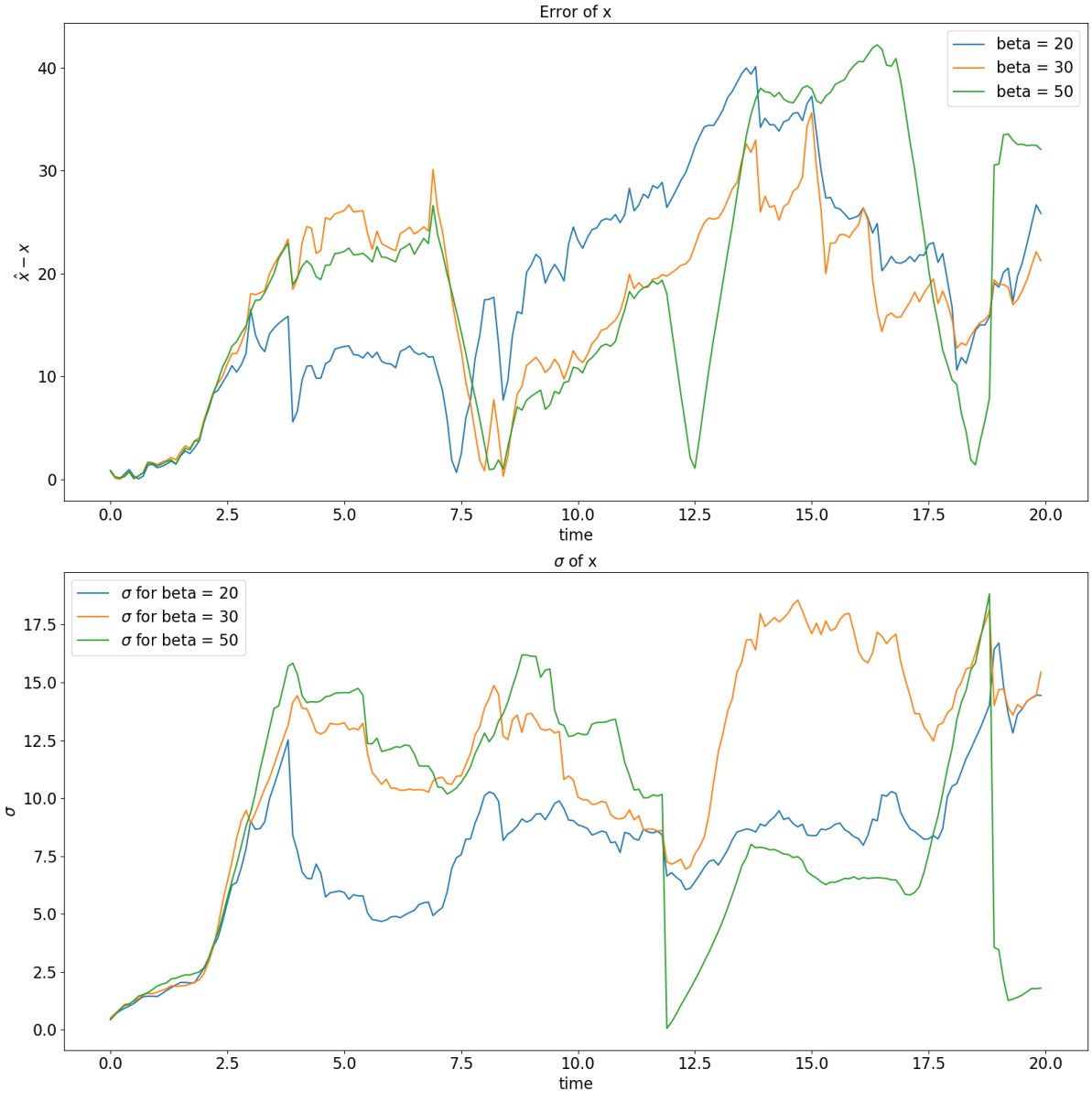

Figure 13: Errors and standard deviation of of x

When we underestimate motion noise we can see grows of the error and dicrease of the covariance for EKF (Fig. 14)

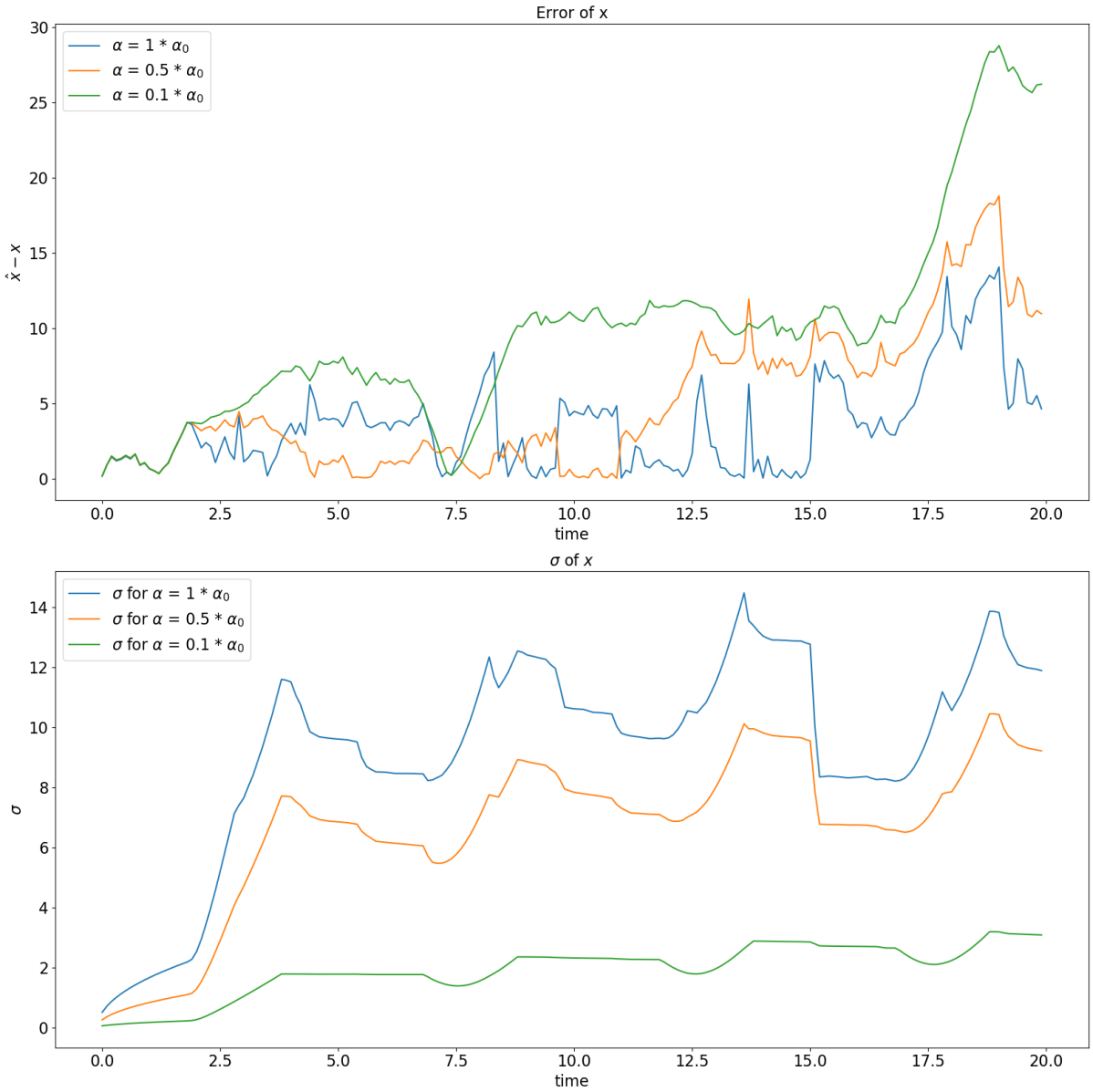

Figure 14: Errors and standard deviation of of x

The same behaviour for the PF (Fig. 15). In general – both filters cannot follow the true path of the robot (it follows ideal trajectory with noise go toward zero)

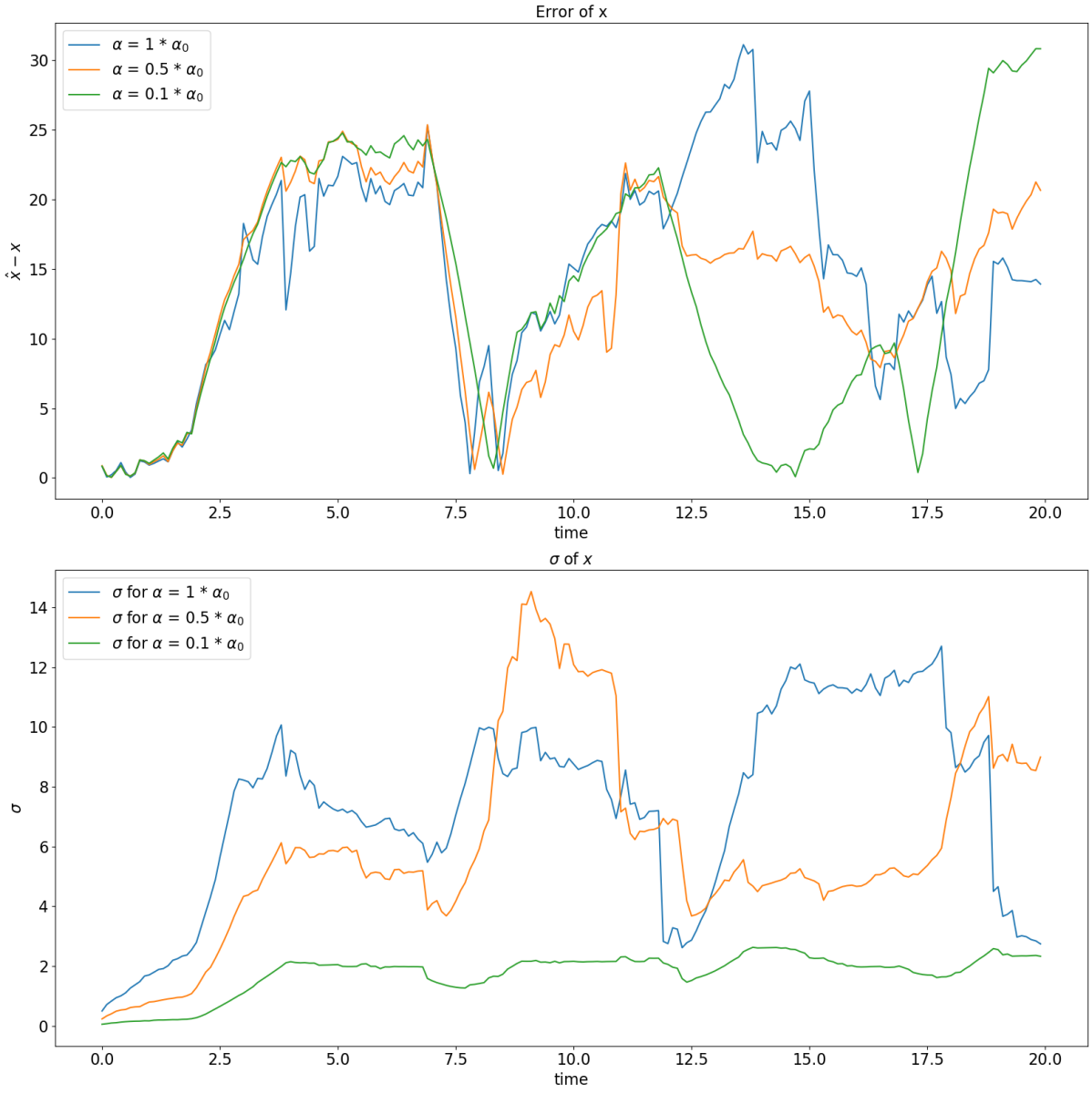

Figure 15: Errors and standard deviation of of x

When you overestimate motion noise, both filters work almost the same, but with greater variance. Behaviour of EKF shown on Fig. 16 and for PF on Fig. 17

Figure 16: Errors and standard deviation of of x

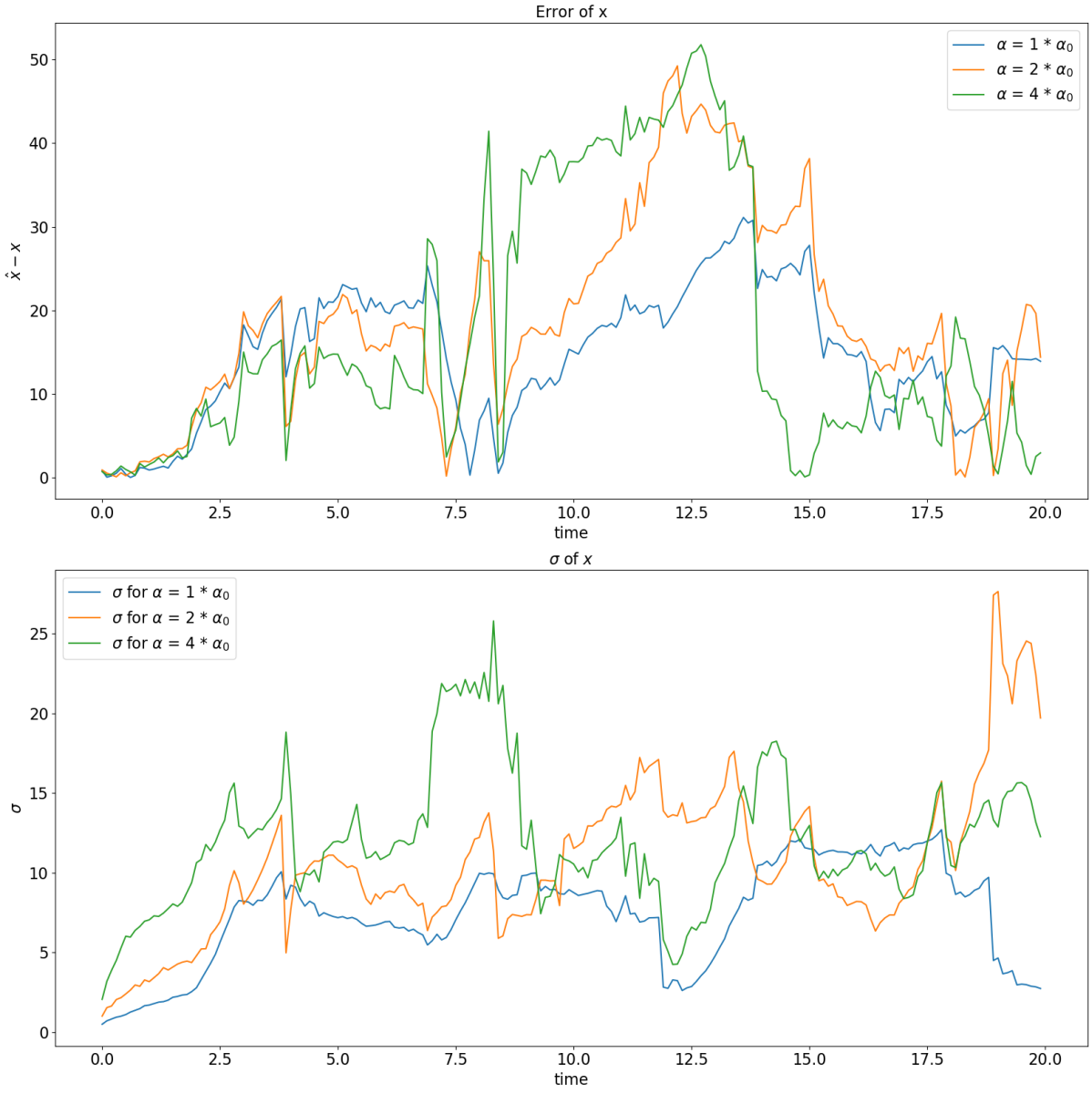

Figure 17: Errors and standard deviation of of x