Introduction

The textwidht for this paper is : 345.0pt
To compile the code

```
cmake .
make
```

To generate the measurements

```
./data gcase00.cfg
```

To run the static parameter case

```
mpirun -np 12 ./run case00.cfg
```

To run the time-varying parameter case

```
mpirun -np 10 ./run case00b.cfg
mpirun -np 10 ./run case00c.cfg
mpirun -np 10 ./run case00d.cfg
mpirun -np 10 ./run case00e.cfg
mpirun -np 10 ./run case00f.cfg
```

To display the results (eps figures will be saved in the figs folder)

```
python draw.py
```

Configuration files

The various parameters can be changed in different files. In the file "functions.cpp", the models are coded. In the ".cfg" files, the parameters for each cases are present. The file "gcase00.cfg" controls data generation. The important parameters are

```
name  =  "First Order SDE (colored noise!)";
handle  = "FirstOrder";
folder = "Case00";
initialState  = [1.0];
parameters = [1.0, 0.01];
time = 10.0;
dt = 0.001;
stepsBetweenMeasurements = 10;
NSR = 0.25;
```

where handle corresponds to the function in "functions.cpp". The **folder** "Case00" is where the measurements will be stored. The initial state of the system is $x = 1.0$ (**initialState**). The parameter vector $[1.0, 0.01]$ corresponds to $[\alpha, \sigma]$. The keyword **time** controls for how long is the system simulated, **dt** is the timestep, **stepsBetweenMeasurements** controls how many time steps between each measurements and **NSR** is the noise to signal ratio.

The other configuration files control the parameter and state estimation problem. In "case00.cfg" the following can be found

```
folder = "Case00";
dt = 0.001;
fStepsBetweenMeasurements  = 10;
measurementCov = "variance.dat";
data = "data.dat";
Parameter_Estimation  = true;
Evidence_Estimation  = false;
State_Estimation = false;
seruns  = 1;
```

Where **folder** specifies where the measurements can bve found, **data** and **measurementCov** are the name of the files where measurements and the measurement variance are stored. The true and false indicates that parameter estimation is performed but stand alone state estimation is not performed. The state estimation procedure included in the parameter estimation problem is still performed. For each proposed models we have the following configuration keywords

```
run = true;
name  = "StaticParameters";
state_estimator = "ekf";
handle  = "m2";
initialState  = [5.0];
nprocs = 1;
initialStateVariance  = ([0.1])
//parameters  = (0.2, 0.1);
prior =  (
"Uniform", 0.0, 10.0,
"Uniform", 0.0, 10.0
);
parallelGroups = 8;
MCMC_CONFIG = {
Method = "TMCMC";
dim = 2;
window = 40000;
};
```

run controls wether this model will be run of not. The keyword **state_estimator** controls what filter is used for state estimation. The **handle** links to the statespace created in ”functions.cpp”. **initialState** and **initialStateVariance** are the initial mean and covariance of the state. We use TMCMC to sample the posterior distribution. The dimension of the pdf is **dim** = 2 and there are window samples divided into 8 **parallelGroups**. Each core is responsible for a group. **nprocs** and **parameters** are not used here.

The other configuration files control the state estimation procedure when the state is augmented. There could only be one configuration file but they are separated for convenience. For example, in case00d.cfg the following can be found

```
run = true;
name  = "TimeVaryingParametersEnkfD";
state_estimator = "enkfmpi";
handle  = "m1enkf";
initialState  = [1.0, 5.0];
nparticles = 10000;
nprocs = 4;
initialStateVariance  = ([0.1, 0.0],[0.0, 0.000001])
parameters  = (0.01,0.01);
```

The statespace used is m1enkf defined in functions.cpp. The filter used is the parallel Enkf (**state_estimator** = ”enkfmpi”) the initial state is controlled by **initialState**. The filter uses 4 processsors (**nprocs**) and is composed of 10000 particles (**nparticles**). The initial state variance (**initialStateVariance**) is

$$P_0 = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.000001 \end{bmatrix} \tag{1}$$

and the parameters for $[\sigma, \gamma]$ are defined by **parameters**. These can be changed as well.

Description of the system

Consider the following linear system

$$\dot{x} + ax = \sigma W(t) \tag{2}$$

with available measurements described by

$$d_k = x_k + \varepsilon_k \tag{3}$$

where ε_k is a GRV with zero mean and a variance chosen so that the noise to signal ratio is 0.25. The discrete representation of this system is given by

$$x_{k+1} = (1.0 - a\Delta t)x_k + \sigma\sqrt{\Delta t}q_k \tag{4}$$

where q_k is drawn from a zero mean and unit variance Gaussian distribution. A realization of this system is generated with a initial condition of $x(0) = 1.0$. Noisy measurements are generated. The measurements are kept dense to insure that the estimation of the time-varying parameter a will succeed with EKF.

The are two approaches considered here to estimate the unknown parameter a. The first method, a is treated as a static parameter. In this approach, both a and σ are estimated.

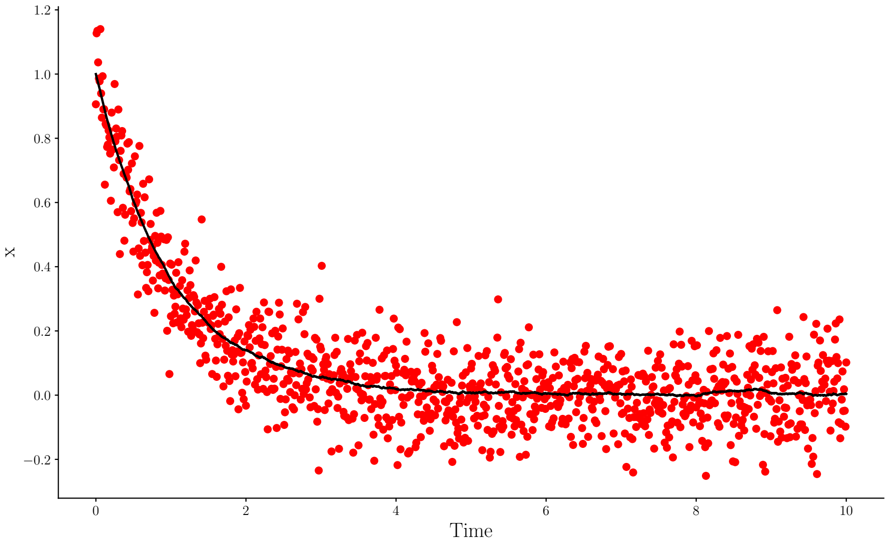

Figure 1: Synthetic measurements

Static parameter

An MCMC algorithm is used to sample from the parameter posterior distribution. In the first case, only the static parameter a is estimated. The marginal pdf of parameter a is shown in Fig. 2. In the second scenario both a and σ are estimated. The marginal pdfs are shown in Fig. 3 and Fig. 4.

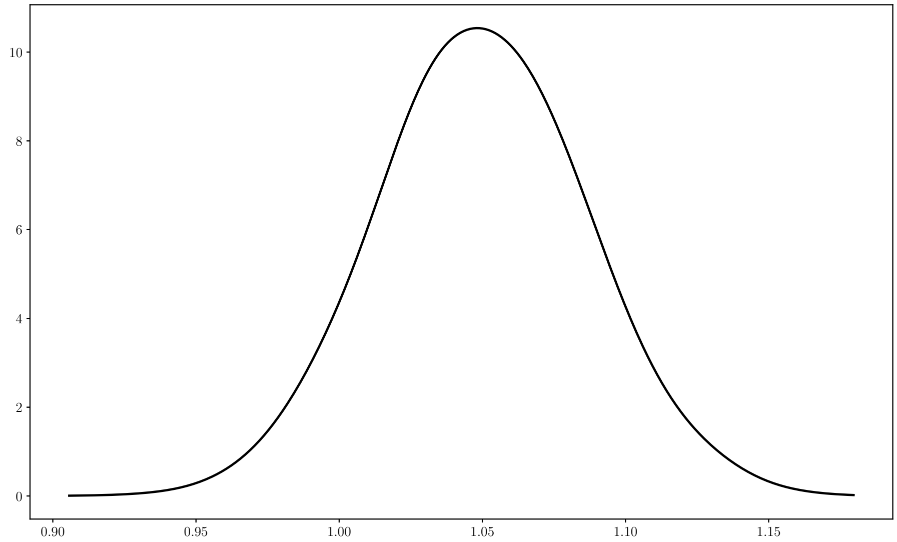

Figure 2: Pdf of static parameter a when it is the only unknown. True value of a is 1.0

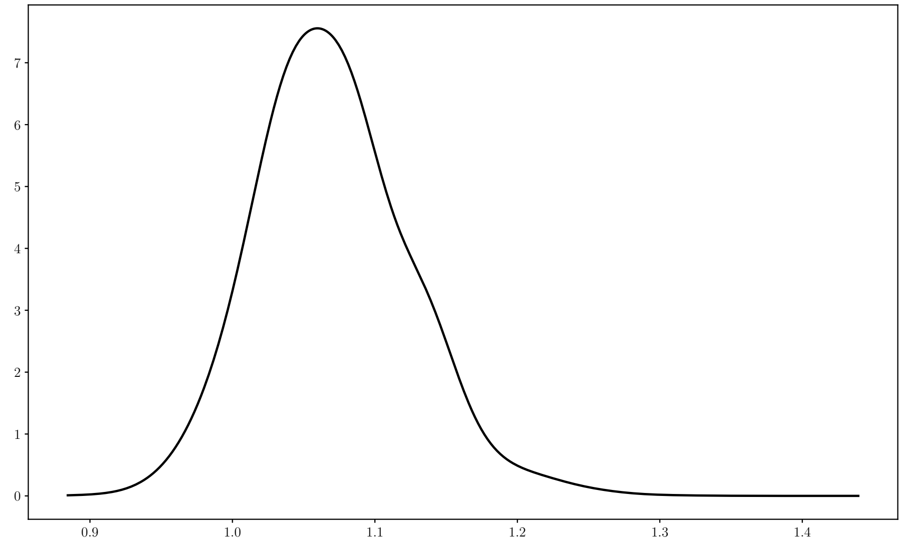

Figure 3: Marginal pdf of static parameter a. True value of a is 1.0

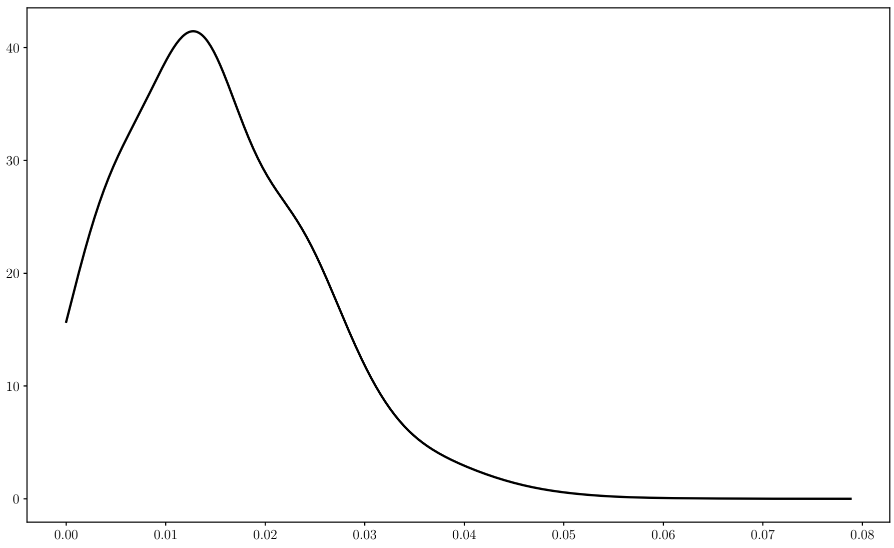

Figure 4: Marginal pdf of static parameter σ. True value of σ is 0.01.

Time-varying parameter

In the second approach, a is treated as a time-varying parameter. The state of the system, fully described by x is augmented with a_k. The original static parameter a is now varying. In the first method, a MCMC algorithm was used to estimate the posterior pdf of the parameters, in the second method, only state estimation is performed. This approach is less expensive, however augmenting the state with the parameters makes the problem nonlinear. State estimation will be performed using EKF, EnKF and PF. The modified discrete system for the first scenario is described by

$$x_{k+1} = (1.0 - a_k \Delta t) x_k + \sigma \sqrt{\Delta t} w_k \tag{5}$$

$$a_{k+1} = a_k + \gamma \sqrt{\Delta t} q_k \tag{6}$$

where γ is a tuning parameter. This value can be seen as a parameter controlling how much the parameter a is allowed to change during the forecast. The strength of the noise is assumed known with $\sigma = 0.01$. The estimate of a_k is updated with available measurements. State estimation is performed using three different filters, EKF, EnKF and PF. Nonlinear filters are necessary due the nonlinear nature of the augmented system. The estimate of x and a are shown in the following figures. The intial variance of the state is set to

$$P_0 = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix} \tag{7}$$

The effect of the initial variance is not studied in this work. But there might be an impact on the quality of the estimates when bad intial estimates are provided. The state estimage of x and a are shown in Fig. 5 and Fig. 6. In this case the initial conditions are $[1.0, 1.2]$. The noise parameters are $\sigma = 0.01$ and $\gamma = 0.01$.

The red dashed lines correspond to 99% bounds of the EKF estimate.

Effect of initial conditions

In this section, the value of the initial state variance, the noise strenght and the initial estimate of a are changed. The different cases are outlined in Table 1.

Table 1: Overview of cases

Cases	a_0	$Var(a_0)$	γ	Figures
Case 1	5.0	0.1	0.01	7,8
Case 2	5.0	0.000001	0.01	9,10
Case 3	5.0	0.000001	2.0	11,12
Case 4	5.0	2.0	0.000001	13,14

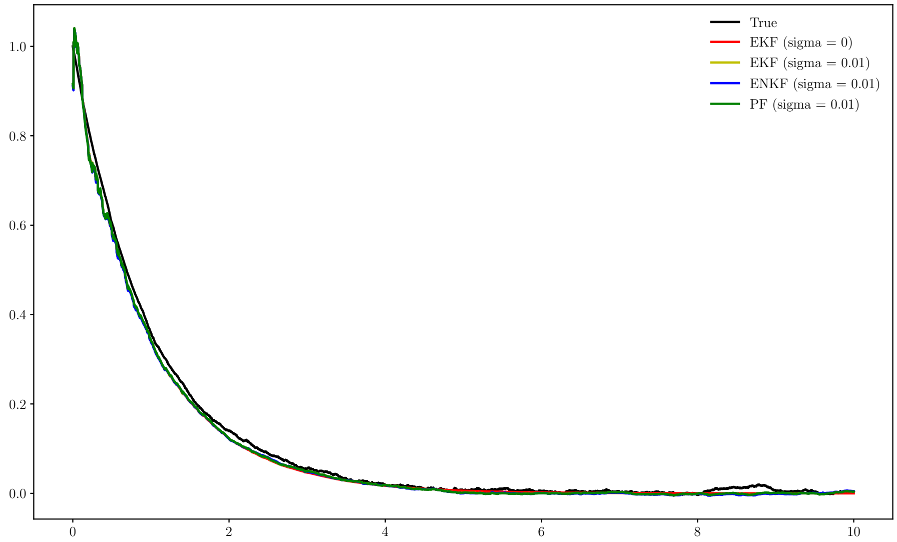

Figure 5: Mean estimate of x when $a = 1.2$.

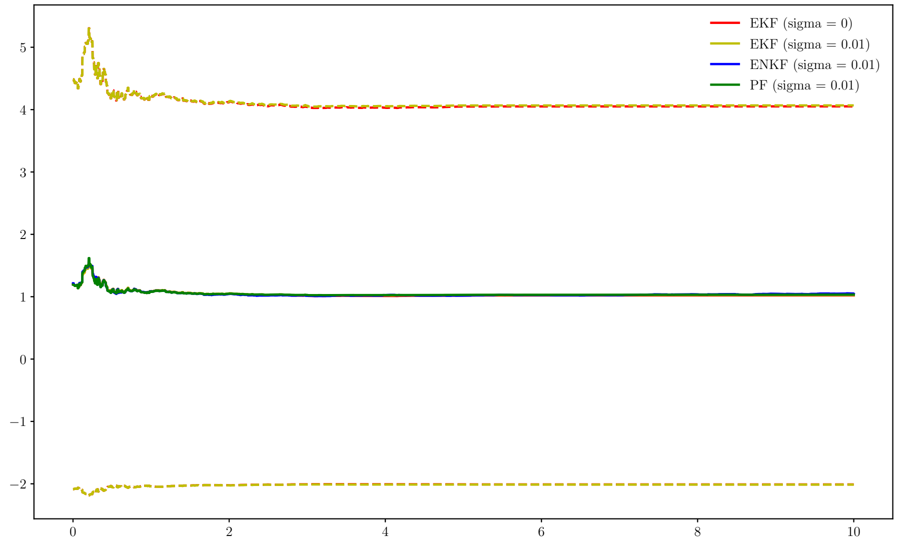

Figure 6: Mean estimate of a when $a = 1.2$.

Performing state estimation by augmenting the state provides the following estimates shown in Fig. 7 and Fig. 8. In this case the initial conditions are $[1.0, 5.0]$. The variance of a_0 is 0.1 and the value of γ is 0.01. In this scenario PF doesn't perform adequately and I do not know why. I've increased the number of particles to 100 000 (about 10 times the number of particles used by EnKF) and still have the same result. I will need to investigate why it doesn't perform as it should. I think part of the reason is that the initial variance for a_0 is to restrictive, it may be that in the update phase no particles have a high loglikelihoood and thus we are running into numerical problems where the weights are too small.

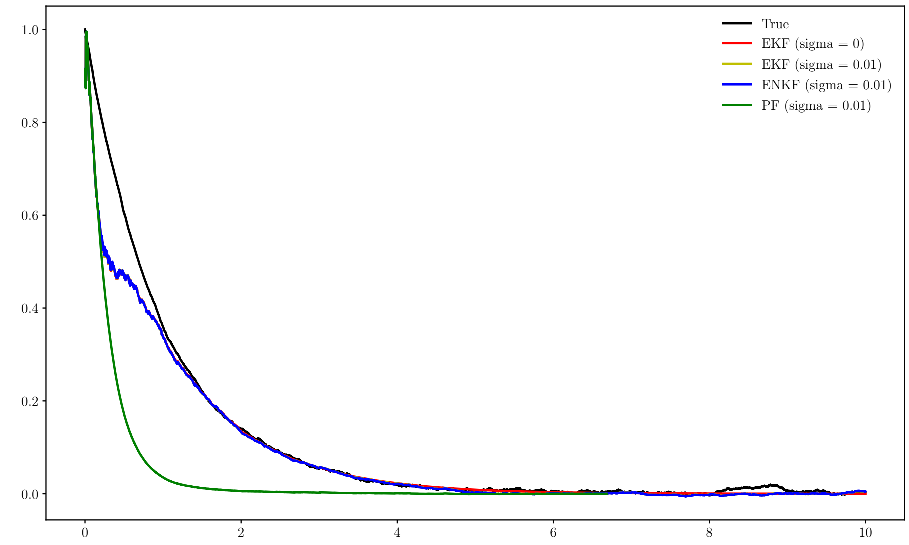

Figure 7: (Case 1) Mean estimate of x.

The results of state estimation for Case 2 are shown in Fig. 9 and Fig. 10. The initial variance of a_0 is significantly decreased. The value of a doesn't change and can't be updated by incoming measurements due to the restrictive nature of the initial condition and the small γ.

The results of state estimation for Case 3 are shown in Fig. 11 and Fig. 12. The initial variance of a_0 is the same as the previous case, but the value of γ is significantly higher. The mean value of a changes but it can't converge to a specific value.

The results of state estimation for Case 4 are shown in Fig. 13 and Fig. 14. The initial variance of a_0 is significantly larger and we have a smaller value of γ. The results are satisfactory because the of larger variance of a. In that case the true value lies in a high probability region of the prior. The effect of γ is not needed.

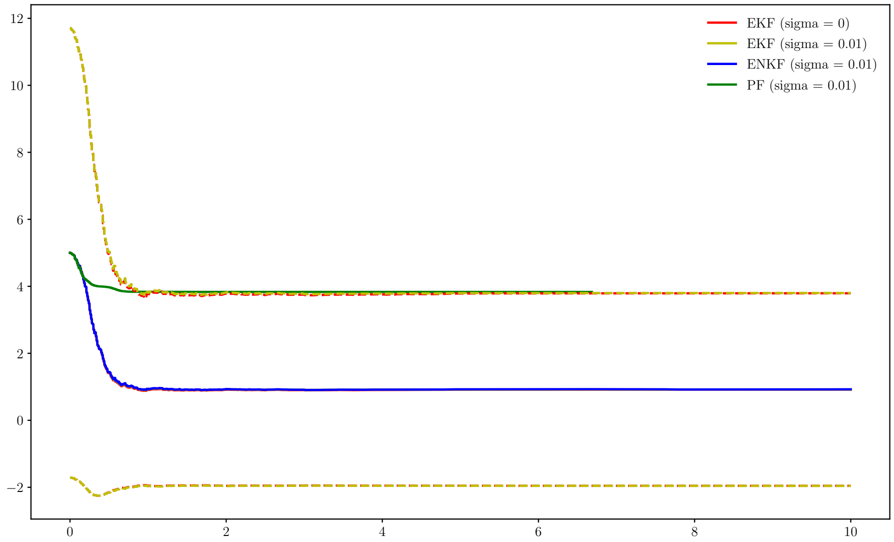

Figure 8: (Case 1) Mean estimate of a.

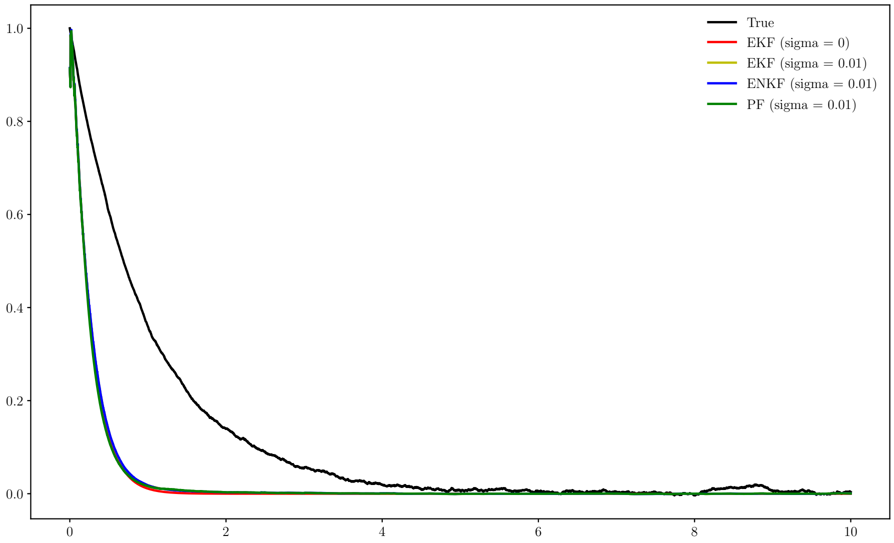

Figure 9: (Case 2) Mean estimate of x.

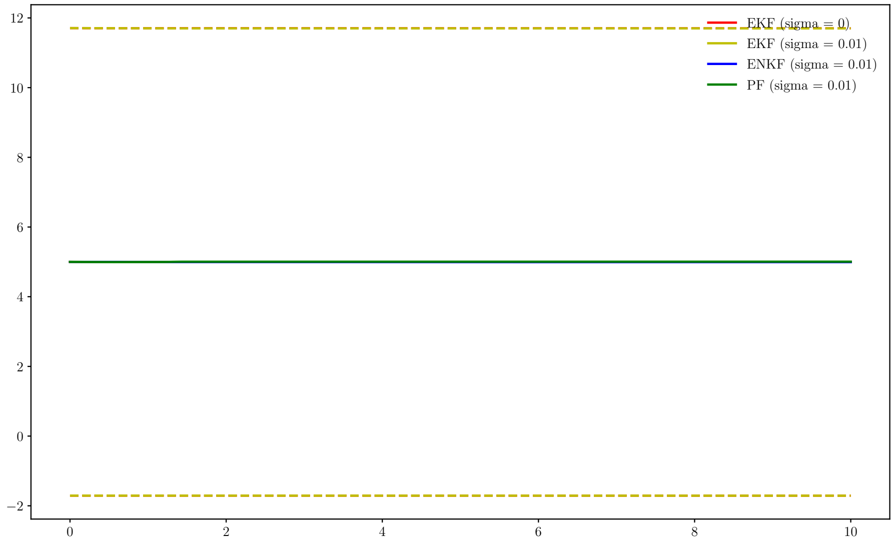

Figure 10: (Case 2) Mean estimate of a.

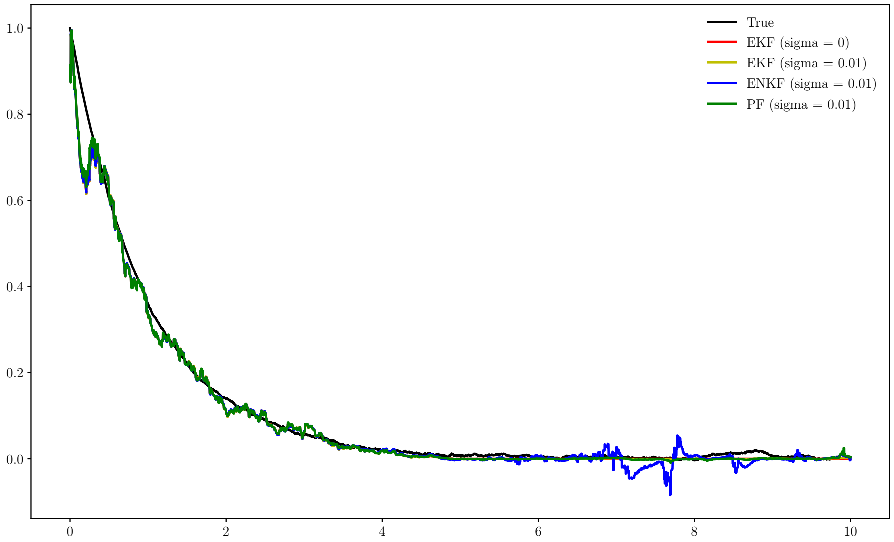

Figure 11: (Case 3) Mean estimate of x.

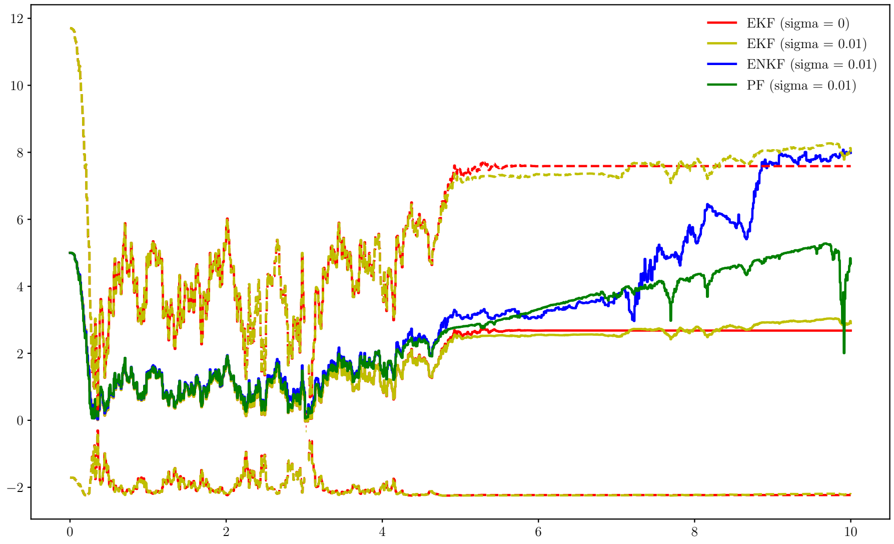

Figure 12: (Case 3) Mean estimate of a.

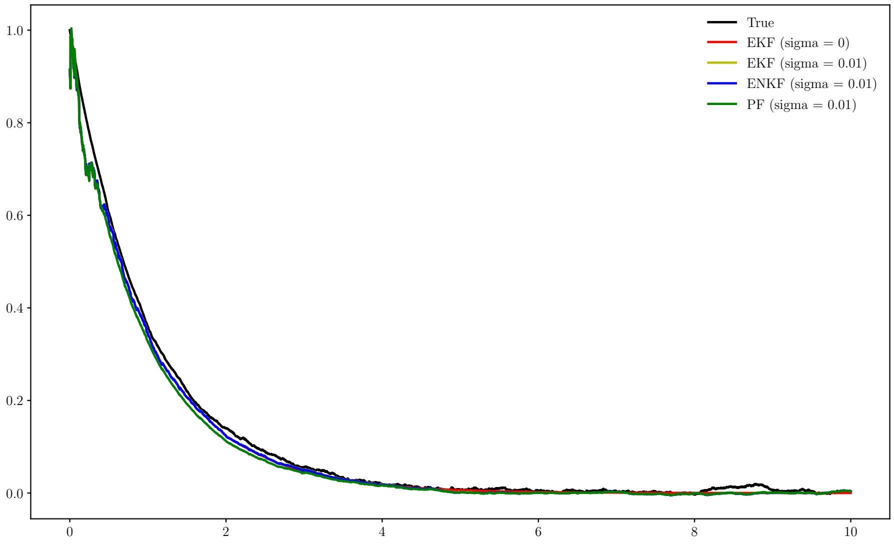

Figure 13: (Case 4) Mean estimate of x.

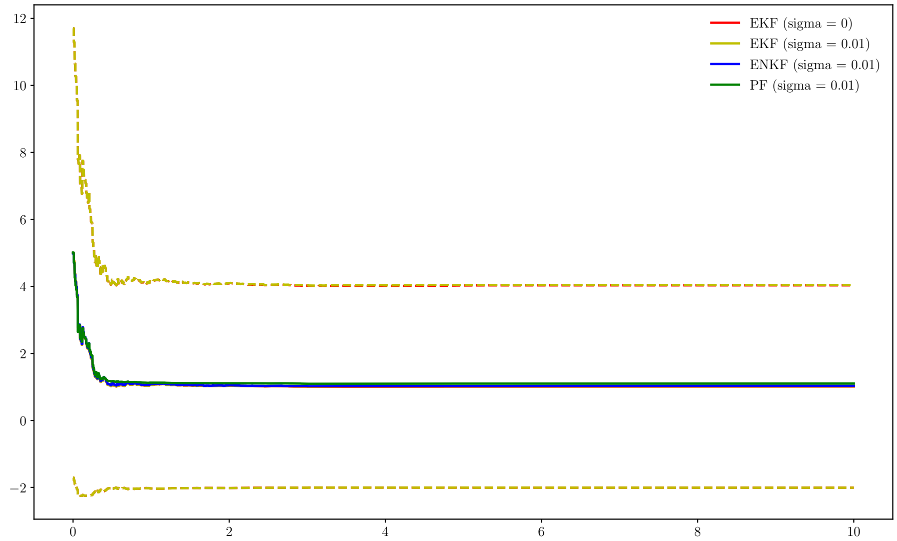

Figure 14: (Case 4) Mean estimate of a.