Laboratory work 2 - Comparison of the exponential and running mean for random walk model

Nikolay Zherdev, Dmitry Baluev, Carolina Latserus

Skoltech, 08.10.2018

The goal of this laboratory work is to compare performance of exponential and running mean for random walk model.

This lab consists of two parts:

I. Determination of optimal smoothing constant in the exponential mean.

II. Comparison of methodical errors of the exponential and running mean.

In [66]:
```
import pandas as pd
import numpy as np
from sklearn.metrics import mean_squared_error
from plotly import __version__
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot
import plotly.graph_objs as go

init_notebook_mode(connected=True)
```

Part 1

1.1.Generate a true trajectory X_i using the random walk model

$$X_i = X_{i-1} + w_i \quad (1)$$

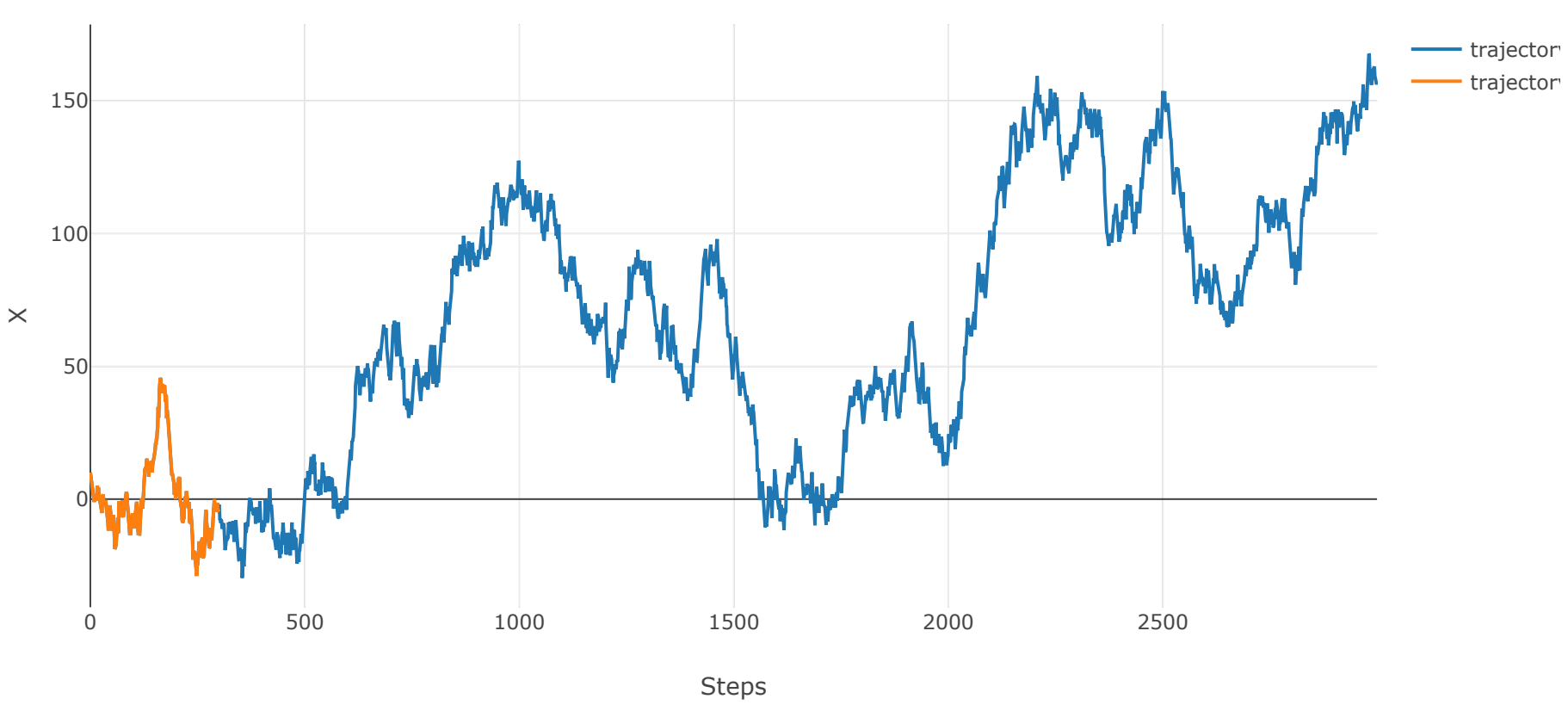
w_i –normally distributed random noise with zero mathematical expectation and variance σ_w^2.
Group 1: $\sigma_w^2 = 8$;
Group 2: $\sigma_w^2 = 9$;

In [67]:
```
def generate_trajectory(x0 = 10, mean = 0, var = 0.1, steps = 100):
    trajectory = np.random.normal(loc = mean, scale = np.sqrt(var), size = steps)
    trajectory[0] = x0
    return np.cumsum(trajectory)
```

In [68]:
```
traj3000 = generate_trajectory(var = 9, steps = 3000)
traj300 = traj3000[:300]
```

In [69]:
```
data = [
    go.Scatter(
        y=traj3000,
        name='trajectory 1'
    ),
    go.Scatter(
        y=traj300,
        name='trajectory 2'
    ),
]

layout= go.Layout(
    title= 'Trajectories',
    xaxis= dict(
        title= 'Steps',
    ),
    yaxis=dict(
        title= 'X',
    ),
    showlegend= True
)
fig= go.Figure(data=data, layout=layout)
iplot(fig)
```

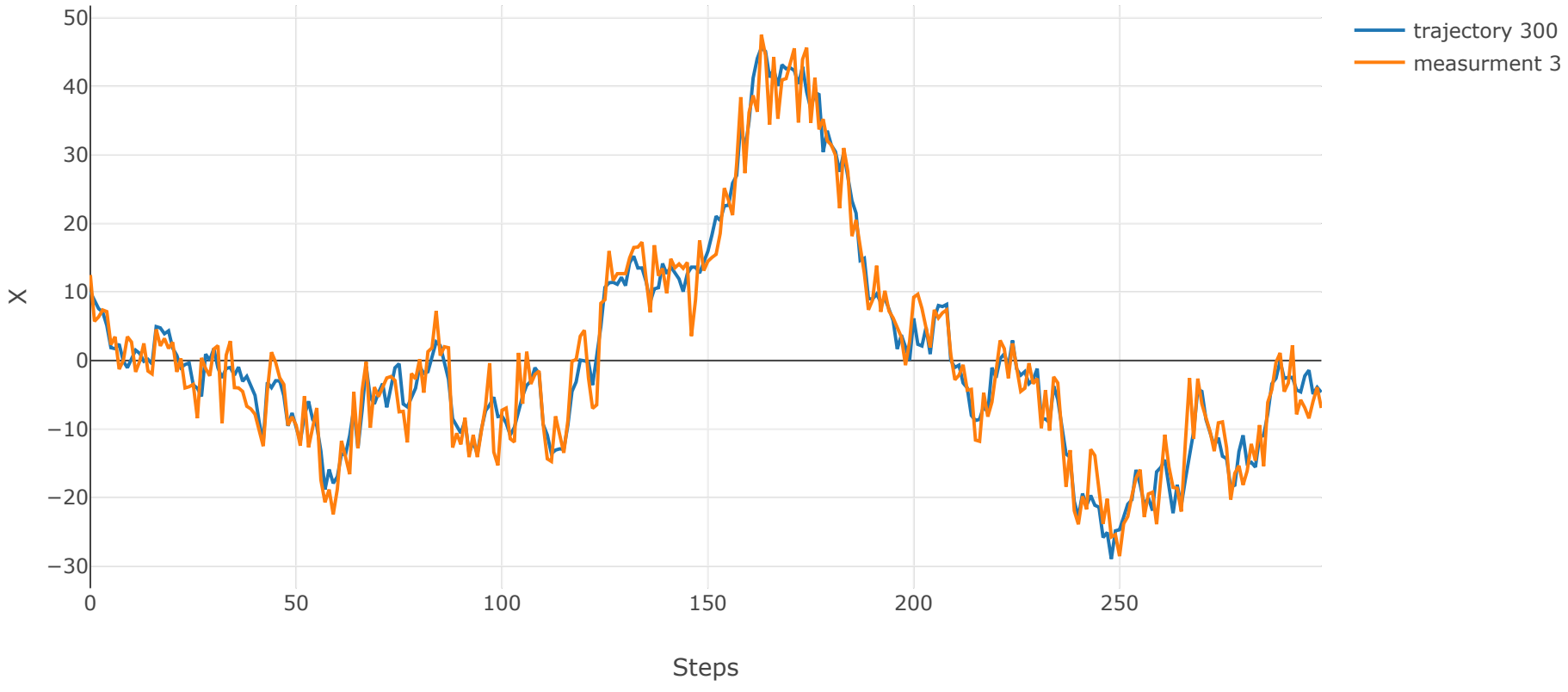


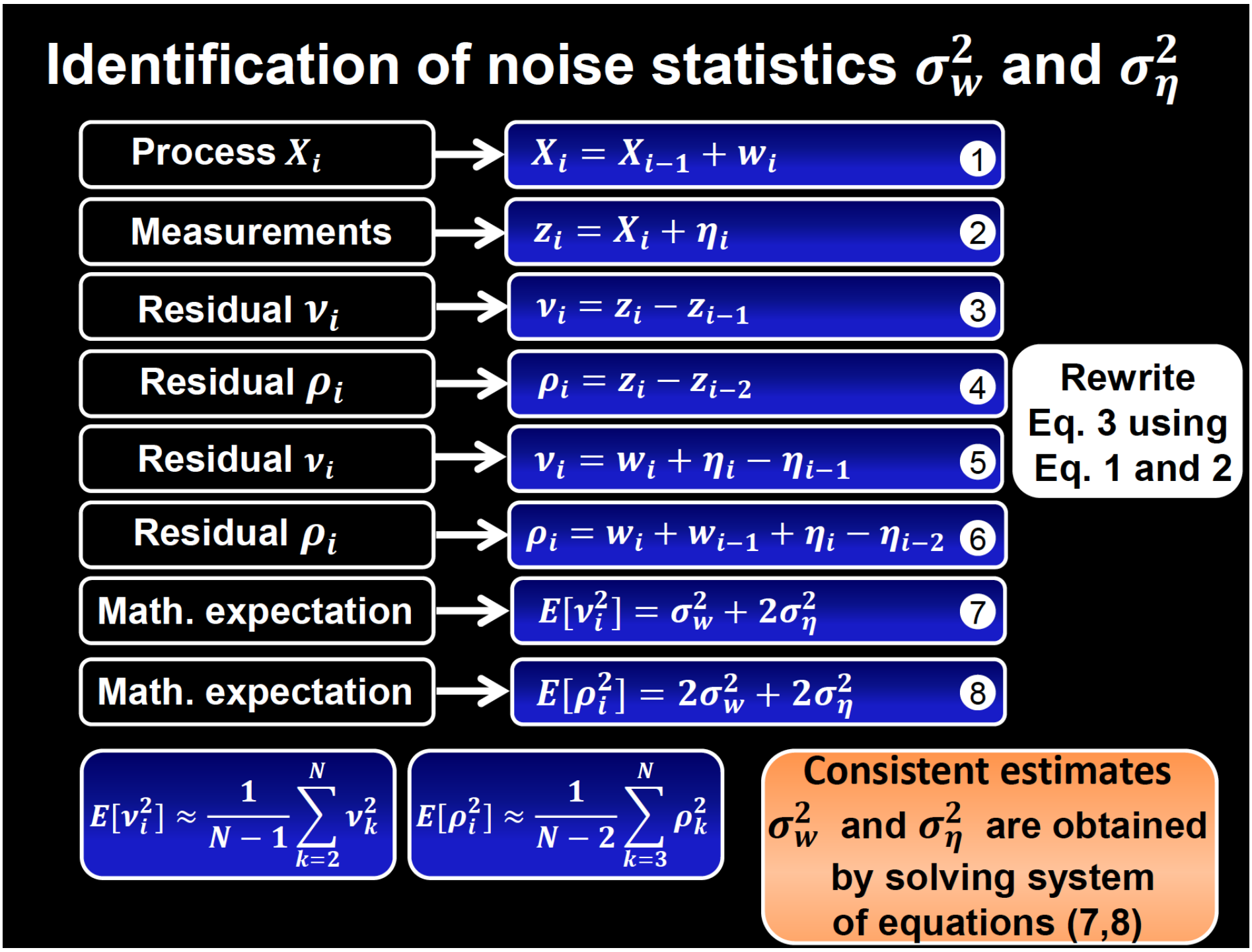
In [70]:
```
def measure(trajectory, mean = 0, var = 0.1):
    noise = np.random.normal(loc = mean, scale = np.sqrt(var), size = trajectory.shape)
    return np.add(trajectory, noise)
```

In [71]:
```
meaz300 = measure(traj300, var = 12)
meaz3000 = measure(traj3000, var = 12)
```

In [72]:
```
data = [
    go.Scatter(
        y=traj300,
        name='trajectory 300'
    ),
    go.Scatter(
        y=meaz300,
        name='measurment 300'
    ),
]

layout= go.Layout(
    title= 'True and measured',
    xaxis= dict(
        title= 'Steps',
    ),
    yaxis=dict(
        title= 'X',
    ),
    showlegend= True
)
fig= go.Figure(data=data, layout=layout)
iplot(fig)
```





2. Identify σ_w^2 and σ_η^2 using identification method presented on slide 55 (Topic_2_Quasi-optimal approximation under uncertainty.pdf). Perform identification for different size of trajectory (3000 and 300). Compare estimation results with true values of σ_w^2 and σ_η^2. Compare the accuracy of estimation.

In [113]:
```
def variations(z):
    vi = np.subtract(z[1:], z[:-1])
    pi = np.subtract(z[2:], z[:-2])
    E_v = np.mean(vi**2)
    E_p = np.mean(pi**2)
    a = np.array([[1, 2],[2, 2]])
    b = np.array([E_v, E_p])
    return np.linalg.solve(a, b) # solve for var_w, var_n

# np.allclose(np.dot(a, x), b)
```

In [114]:
```
var_w, var_n = variations(meaz300)
print("var_w is", float("{:.2f}".format(var_w)))
print("var_n is", float("{:.2f}".format(var_n)))
print("ref values are", 9, "and", 12, "correspondingly")
```

```
var_w is 9.58
var_n is 8.19
ref values are 9 and 12 correspondingly
```

In [115]:
```
var_w, var_n = variations(meaz3000)
print("var_w is", float("{:.2f}".format(var_w)))
print("var_n is", float("{:.2f}".format(var_n)))
print("ref values are", 9, "and", 12, "correspondingly")
```

```
var_w is 9.52
var_n is 11.87
ref values are 9 and 12 correspondingly
```

On a small dataset the estimated variances differ greatly from table values, but for increased dataset their values became much closer to the real ones.

3. Determine optimal smoothing coefficient in exponential smoothing

$$\alpha = \frac{-\chi+\sqrt{\chi^2+4\chi}}{2} \qquad (3)$$

$$\chi = \frac{\sigma_w^2}{\sigma_\eta^2}$$

In [116]:
```
def smoothing_coeff(var_w, var_n):
    xi = var_w / var_n
    return (-xi + np.sqrt(xi**2 + 4*xi))/2
```

In [117]:
```
opt_alpha = smoothing_coeff(var_w, var_n)
print("optimum alpha is", float("{:.2f}".format(opt_alpha)))
```

```
optimum alpha is 0.58
```

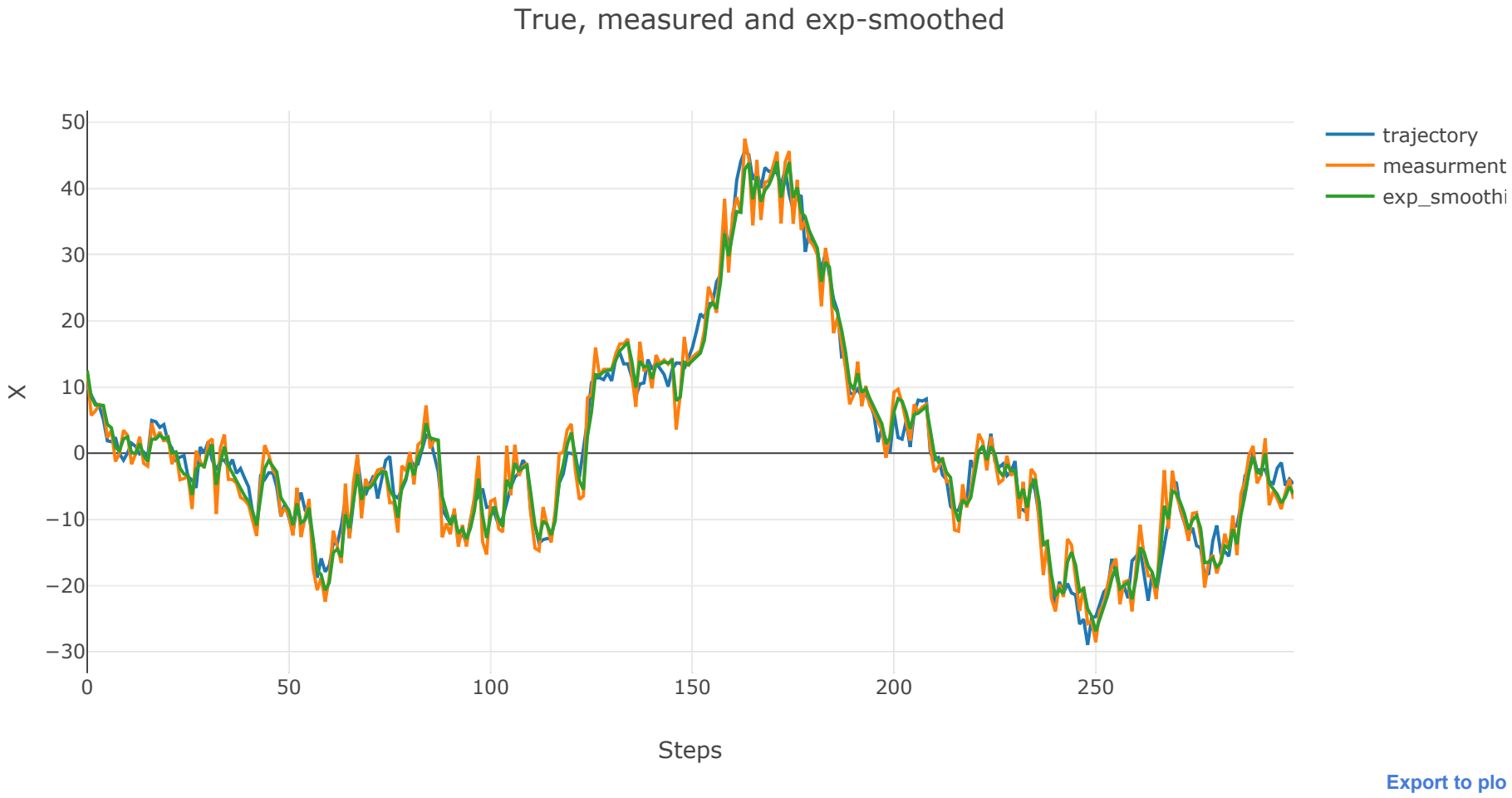
4. Perform exponential smoothing with the determined smoothing coefficient. Plot results. For the comparison add measurements, true values of process and exponentially smoothed data.

In [78]:
```
def exp_smooth(z, alpha):
    X = np.zeros(z.shape)
    X[0] = z[0]
    for i in range(1, z.shape[0]):
        X[i] = X[i-1] + alpha*(z[i] - X[i-1])
    return X
```

In [79]:
```
smooth300 = exp_smooth(meaz300, opt_alpha)
```

In [105]:
```
data = [
    go.Scatter(
        y=traj300,
        name='trajectory'
    ),
    go.Scatter(
        y=meaz300,
        name='measurment'
    ),
    go.Scatter(
        y=smooth300,
        name='exp_smoothing'
    ),
]

layout= go.Layout(
    title= 'True, measured and exp-smoothed',
    xaxis= dict(
        title= 'Steps',
    ),
    yaxis=dict(
        title= 'X',
    ),
    showlegend= True
)
fig= go.Figure(data=data, layout=layout)
iplot(fig)
```



Part 2

Comparison of methodical errors of exponential and running mean.

1. Generate a true trajectory X_i using the random walk model (1).
   Size of trajectory is 300 points.
   Initial condition $X_1 = 10$.
   Variance of noise w_i, $\sigma_w^2 = 28^2$

In [118]:
```
trajectory = generate_trajectory(x0 = 10, var = 28**2, steps = 300)
```

2. Generate measurements z_i of the process X_i using equation (2)
   Variance of noise measurement noise η_i, $\sigma_\eta^2 = 97^2$

In [119]:
```
measurments = measure(trajectory, var = 97**2)
```

3. Determine optimal smoothing coefficient α using equation (3).
(There is no need to identify it again, just use equation for α from part I).

In [122]:
```
var_w, var_n = 28**2, 97**2

# apply EM with alpha to measurments
alpha = smoothing_coeff(var_w, var_n)
exp_smoothed = exp_smooth(measurments, alpha)
print("optimum alpha is", float("{:.2f}".format(alpha)))
```

```
optimum alpha is 0.25
```

4. The component of full error that is related to measurement errors is determined as (from slide 37, Topic_2_Quasi-optimal approximation under uncertainty.pdf)

Running mean (RM):

$$\sigma_{RM}^2 = \frac{\sigma_\eta^2}{M} \quad (4)$$

In [123]:
```
M = (2-alpha)/alpha
```

In [124]:
```
M
```

Out[124]:
```
7.000364422000923
```

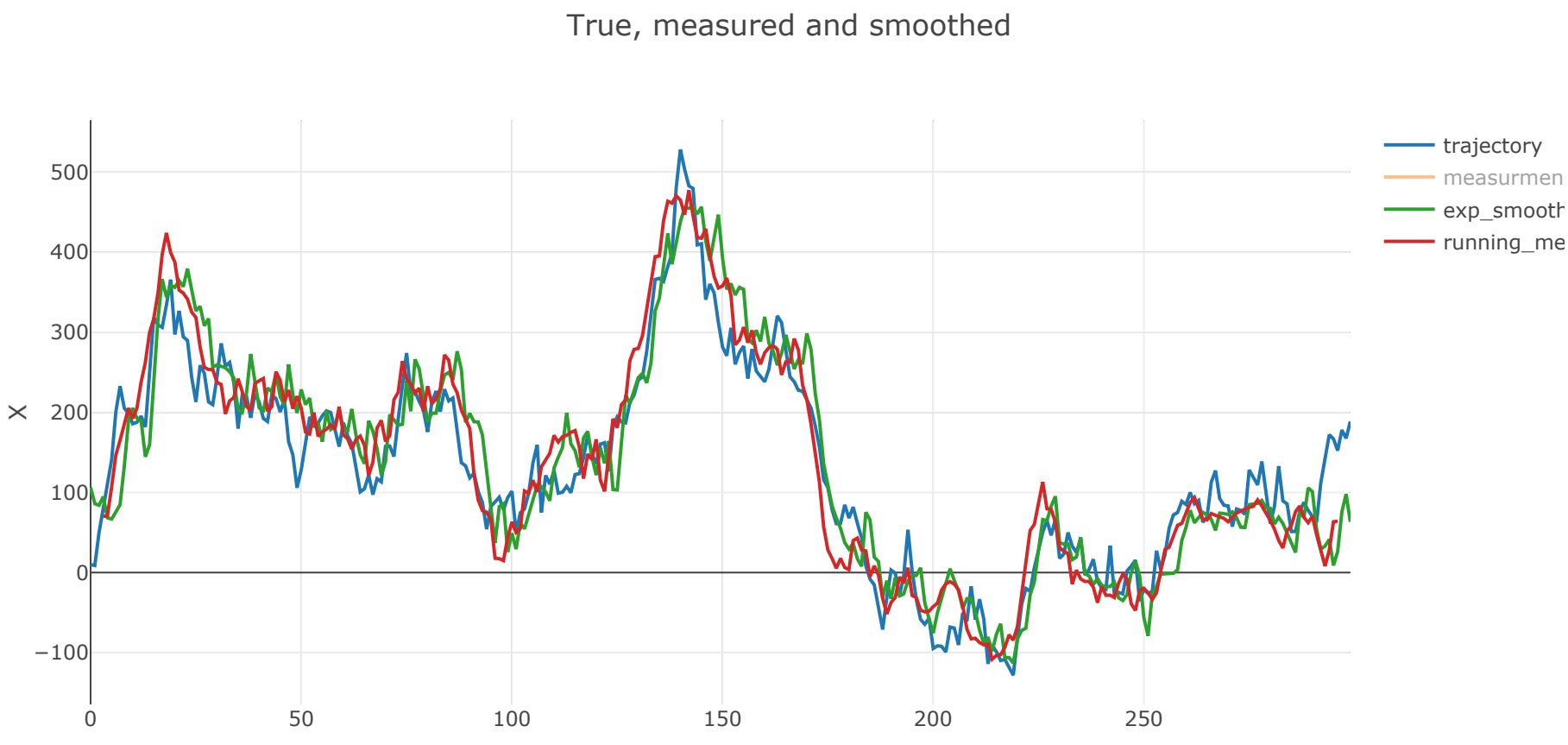
5. Apply running mean using determined window size M and exponential mean (see, for instance, page 30, Topic_2_Quasi-optimal approximation under uncertainty.pdf) using determined smoothing constant α to measurements z_i. Plot true trajectory X_i, measurements z_i, running and exponential mean.

In [125]:
```
# apply RM with window M to measurments

running_mean = pd.Series(measurments).rolling(window=int(M), center = True).mean()
```

In [126]:
```
data = [
    go.Scatter(
        y=trajectory,
        name='trajectory'
    ),
    go.Scatter(
        y=measurments,
        name='measurments'
    ),
    go.Scatter(
        y=exp_smoothed,
        name='exp_smoothed'
    ),
    go.Scatter(
        y=running_mean,
        name='running_mean'
    ),
]

layout= go.Layout(
    title= 'True, measured and smoothed',
    xaxis= dict(
        title= 'Steps',
    ),
    yaxis=dict(
        title= 'X',
    ),
    showlegend= True
)
fig= go.Figure(data=data, layout=layout)
iplot(fig)
```



6. Make visual comparison of results. Make conclusions which methods give greater methodical error in conditions of equal errors conditioned by measurement errors for this particular generated trajectory.

In [127]:
```
# Determine the variance of deviation of smoothed data from the true one.
RM_error = running_mean - trajectory # running mean
ES_error = exp_smoothed - trajectory # exp smoothing

RM_var = np.var(RM_error, ddof = 1)
ES_var = np.var(ES_error, ddof = 1)

print("Running mean variance", float("{:.2f}".format(RM_var)))
print("Exp smoothing variance", float("{:.2f}".format(ES_var)))
```

```
Running mean variance 1873.31
Exp smoothing variance 2412.75
```

It was concluded that the method of exponential smoothing gives bigger time delay, than the method of running mean. By calculating variances of both methods we obtained exact result: Running mean performs more accurately.