Exercise_round_4

February 6, 2025

1 Exercise 1. (Extended Kalman Filter)

Consider the following non-linear state space model:

$$x_k = x_{k-1} - 0.01\sin(x_{k-1}) + q_{k-1}$$

$$y_k = 0.5\sin(2x_k) + r_k$$

where $q_{k-1} \sim N(0, 0.01^2)$ and $r_k \sim N(0, 0.02)$. Derive the required derivatives for an EKF and implement the EKF for the model. Simulate trajectories from the model, compute the RMSE values, and plot the result.

(Note that Exercise 1 does not come with a template, so feel free to use whatever framework you want.)

[1]:
```
import matplotlib.pyplot as plt
import numpy as np
import scipy.linalg as linalg
```

Let's do a test simulation of the state space model first.

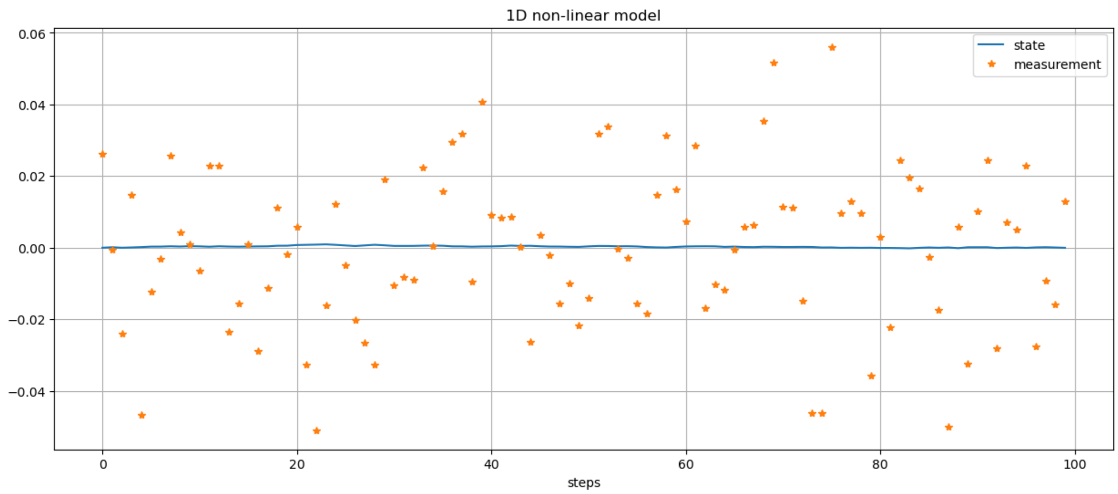
[2]:
```
def model_simulation(steps):
    """
    1D non-linear model simulation
    -------
    Input:
        seed_number: it is used to generate the same sequence of random numbers
        steps: number of steps
    Output:
        xs: state trajectory
        ys: measurement tajectory

    """
    xs = np.zeros((steps, 1))
    ys = np.zeros((steps, 1))

    for i in range(steps):
        xs[i] = xs[i-1] - 0.01*np.sin(xs[i-1]) + np.random.normal(0,0.01**2)
        ys[i] = 0.5*np.sin(2*xs[i]) + np.random.normal(0,0.02)
```

```
    return xs, ys
```

[3]:
```
xs, ys = model_simulation(100)
plt.figure(figsize=(15,6))
plt.plot(xs, label='state')
plt.plot(ys, '*', label='measurement')
plt.title('1D non-linear model')
plt.xlabel('steps')
plt.legend()
plt.grid();
```



Let's implement the Jacobian functions F_x and H_x.

[4]:
```
def F_x(x):

    """
    Derivative of dynamic functin
    -------
    Input:
        x: state
    Output:
        Fx: value of derivative of the dynamic function tanh(.) at state x

    """
    Fx = 1 - 0.01*np.cos(x)

    return Fx

def H_x(x):
```

```
    """
    Derivative of the measurement function
    -------
    Input:
        x: state
    Output:
        Hx: value of derivative of the measurement function sin(.) at state x

    """

    Hx = np.cos(2*x)

    return Hx
```

Implementation of the main EKF function.

[5]:
```
def Extended_Kalman_Filter(Y):
    """
    Extended Kalman filter state estimation for 1D non-linear state space model

    -------
    Input:
        Y: measurements
    Output:
        mean_ekf: Extended Kalman filter mean estimation
        cov_ekf: Extended Kalman filter covariance estimation

    """
    steps = Y.shape[0]
    mean_ekf = np.zeros((steps, 1))
    cov_ekf = np.zeros((steps, 1, 1))

    cov_ekf[0] = 0.1

    Q = 0.01**2
    R = 0.02

    for n in range(steps-1):
        # Prediction
        x_ = mean_ekf[n] - 0.01*np.sin(mean_ekf[n])
        Fx = F_x(mean_ekf[n])
        P_ = Fx**2 * cov_ekf[n] + Q

        # Update
        Hx = H_x(x_)
        S = Hx**2*P_ + R
        K = P_ * Hx * (1/S)
```

```
        mean_ekf[n+1] = x_ + K * (Y[n]-0.5*np.sin(2*x_))
        cov_ekf[n+1] = P_ - K*Hx*P_

    return mean_ekf, cov_ekf
```

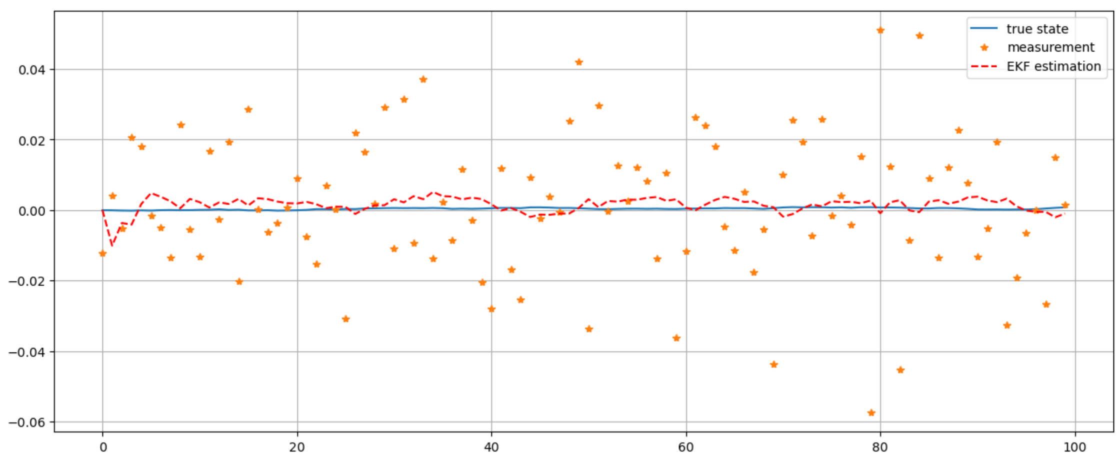
Let's do a test run of the filter!

[6]:
```
states, observations = model_simulation(100)
x_ekf, cov_ekf = Extended_Kalman_Filter(observations)
plt.figure(figsize=(15,6))
plt.plot(states, label='true state')
plt.plot(observations, '*', label='measurement')
plt.plot(x_ekf[:,0], 'r--', label='EKF estimation')
plt.legend()
plt.grid();

# Compute RMSE
rmse_raw = np.sqrt(np.mean((observations - states) ** 2))
rmse_ekf = np.sqrt(np.mean((x_ekf[:, 0] - states) ** 2))

# Print RMSE values
print(f"RMSE (measurement vs state): {rmse_raw:.4f}")
print(f"RMSE (EKF vs state): {rmse_ekf:.4f}")
```

```
RMSE (measurement vs state): 0.0198
RMSE (EKF vs state): 0.0024
```



2 Exercise 2. (Single Iteration of IEKF)

Show that the iterated extended Kalman filter with a single iteration is equivalent to the first order (non-iterated) extended Kalman filter.

⇒ The prediction and update equations for the EFK are as follows.

$$\mathbf{m}_k^- = \mathbf{f}(\mathbf{m}_{k-1})$$

$$\mathbf{P}_k^- = \mathbf{F}_\mathbf{x}(\mathbf{m}_{k-1})\mathbf{P}_{k-1}\mathbf{F}_\mathbf{x}^\top(\mathbf{m}_{k-1}) + \mathbf{Q}_{k-1}$$

$$\mathbf{v}_k = \mathbf{y}_k - \mathbf{h}(\mathbf{m}_k^-)$$

$$\mathbf{S}_k = \mathbf{H}_\mathbf{x}(\mathbf{m}_k^-)\mathbf{P}_k^-\mathbf{H}_\mathbf{x}^\top(\mathbf{m}_k^-) + \mathbf{R}_k$$

$$\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}_\mathbf{x}^\top(\mathbf{m}_k^-)\mathbf{S}_k^{-1}$$

$$\mathbf{m}_k = \mathbf{m}_k^- + \mathbf{K}_k\mathbf{v}_k$$

$$\mathbf{P}_k = \mathbf{P}_k^- - \mathbf{K}_k\mathbf{S}_k\mathbf{K}_k^\top$$

The prediction equations are same for EKF and IEKF. The update equations for the IEKF are a bit different.

For $i = 1, 2, ..., I_{max}$ and the initial guess $\mathbf{x}_k^{(0)} = \mathbf{m}_k^-$, we have:

$$\mathbf{v}_k^{(i)} = \mathbf{y}_k - \mathbf{h}(\mathbf{x}_k^{(i-1)}) - \mathbf{H}_\mathbf{x}(\mathbf{x}_k^{(i-1)})(\mathbf{m}_k^- - \mathbf{x}_k^{(i-1)})$$

$$\mathbf{S}_k^{(i)} = \mathbf{H}_\mathbf{x}(\mathbf{x}_k^{(i-1)})\mathbf{P}_k^-\mathbf{H}_\mathbf{x}^\top(\mathbf{x}_k^{(i-1)}) + \mathbf{R}_k$$

$$\mathbf{K}_k^{(i)} = \mathbf{P}_k^-\mathbf{H}_\mathbf{x}^\top(\mathbf{x}_k^{(i-1)})\left(\mathbf{S}_k^{(i)}\right)^{-1}$$

$$\mathbf{x}_k^{(i)} = \mathbf{m}_k^- + \mathbf{K}_k^{(i)}\mathbf{v}_k^{(i)}$$

$$\mathbf{m}_k = \mathbf{x}_k^{(I_{max})}$$

$$\mathbf{P}_k = \mathbf{P}_k^- - \mathbf{K}_k^{(I_{max})}\mathbf{S}_k^{(I_{max})}\left(\mathbf{K}_k^{(I_{max})}\right)^\top$$

Now for the first iteration of the IEKF, i.e. when $i = 1$, we get:

$$\mathbf{v}_k^{(1)} = \mathbf{y}_k - \mathbf{h}(\mathbf{x}_k^{(0)}) - \mathbf{H}_{\mathbf{x}}(\mathbf{x}_k^{(0)})(\mathbf{m}_k^- - \mathbf{x}_k^{(0)})$$

$$\mathbf{S}_k^{(1)} = \mathbf{H}_{\mathbf{x}}(\mathbf{x}_k^{(0)})\mathbf{P}_k^- \mathbf{H}_{\mathbf{x}}^\top(\mathbf{x}_k^{(0)}) + \mathbf{R}_k$$

$$\mathbf{K}_k^{(1)} = \mathbf{P}_k^- \mathbf{H}_{\mathbf{x}} \top (\mathbf{x}_k^{(0)}) \left(\mathbf{S}_k^{(1)}\right)^{-1}$$

$$\mathbf{x}_k^{(1)} = \mathbf{m}_k^- + \mathbf{K}_k^{(1)} \mathbf{v}_k^{(1)}$$

Substituting $\mathbf{x}_k^{(0)} = \mathbf{m}_k^-$, the equations become

$$\mathbf{v}_k^{(1)} = \mathbf{y}_k - \mathbf{h}(\mathbf{m}_k^-)$$

$$\mathbf{S}_k^{(1)} = \mathbf{H}_{\mathbf{x}}^\top(\mathbf{m}_k^-)\mathbf{P}_k^- \mathbf{H}_{\mathbf{x}}^\top(\mathbf{m}_k^-) + \mathbf{R}_k$$

$$\mathbf{K}_k^{(1)} = \mathbf{P}_k^- \mathbf{H}_{\mathbf{x}}^\top(\mathbf{m}_k^-) \left(\mathbf{S}_k^{(1)}\right)^{-1}$$

$$\mathbf{x}_k^{(1)} = \mathbf{m}_k^- + \mathbf{K}_k^{(1)} \mathbf{v}_k^{(1)}$$

And since $I_{max} = 1$, we can replace the final state update and set the updated covariance as

$$\mathbf{m}_k = \mathbf{m}_k^- + \mathbf{K}_k^{(1)} \mathbf{v}_k^{(1)}$$

$$\mathbf{P}_k = \mathbf{P}_k^- - \mathbf{K}_k^{(1)} \mathbf{S}_k^{(1)} \left(\mathbf{K}_k^{(1)}\right)^\top$$

And thus we can see that the first iteration of IEKF is the same as first order EKF.

See `Exercise7_2.ipynb` for exercise 3.