complex_navigation_sensor_fusion_bayesian_net_perfect

December 26, 2025

[1]:
```
from pathlib import Path
from scipy.io import loadmat
import sys
import os

notebook_path = os.getcwd()
print (f"Current notebook path: {notebook_path}")
project_root = os.path.dirname(notebook_path)
if project_root not in sys.path:
    sys.path.insert(0, project_root)
print (f"Added {project_root} to sys.path")
```

```
Current notebook path: /home/luky/skola/KalmanNet-for-state-
estimation/navigation NCLT dataset
Added /home/luky/skola/KalmanNet-for-state-estimation to sys.path
```

[2]:
```
import torch
import matplotlib.pyplot as plt
from utils import trainer
from utils import utils
from Systems import DynamicSystem
import Filters
import torch.nn.functional as F
from torch.utils.data import TensorDataset, DataLoader
import numpy as np
from scipy.io import loadmat
from scipy.interpolate import RegularGridInterpolator
import random

torch.manual_seed(42)
np.random.seed(42)
random.seed(42)
if torch.cuda.is_available():
    torch.cuda.manual_seed_all(42)

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
DEVICE = device  # For backward compatibility
print(f"device: {device}")
```

device: cuda

[3]:
```
import torch
import torch.nn as nn
import torch.nn.functional as F
from torch.utils.data import TensorDataset, DataLoader
import numpy as np
import matplotlib.pyplot as plt
import os
import Systems


# Parametry sekvencí
TRAIN_SEQ_LEN = 50       # Délka sekvence pro trénink (např. 100 kroků = 100␣
 ↪sekund při 1Hz)
VAL_SEQ_LEN = 150
TEST_SEQ_LEN = 1000      # Délka sekvence pro testování (delší sekvence pro␣
 ↪stabilnější vyhodnocení)
STRIDE = 20          # Posun okna (překryv) pro data augmentation
BATCH_SIZE = 256
DATA_PATH = 'data/processed'
print(f"Běží na zařízení: {device}")
```

Běží na zařízení: cuda

[4]:
```
import torch
from torch.utils.data import TensorDataset, DataLoader
import os

def prepare_sequences(dataset_list, seq_len, stride, mode='train'):
    """
    Zpracuje list trajektorií na sekvence pro trénink dle článku.

    Nový formát dle [Song et al., 2024]:
    - Vstup u (4D): [v_left, v_right, theta_imu, omega_imu]
    - Cíl x (6D):   [px, py, vx, vy, theta, omega]
    """
    X_seq_list = [] # Ground Truth (Cíl)
    Y_seq_list = [] # GPS Měření (Vstup do korekce)
    U_seq_list = [] # Control Input (IMU/Odo)

    print(f"Zpracovávám {len(dataset_list)} trajektorií pro {mode}...")

    for traj in dataset_list:
        # 1. Extrahuje data
        # GT z preprocessingu je [px, py, theta]
        gt = traj['ground_truth'].float()
```

```
        # GPS: [x, y] (obsahuje NaN!)
        gps = traj['filtered_gps'].float()

        # IMU: [ax, ay, theta, omega]
        imu = traj['imu'].float()
        theta_imu = imu[:, 2] # Orientace z IMU
        omega_imu = imu[:, 3] # Úhlová rychlost z IMU

        # ODO: [v_left, v_right]
        odo = traj['filtered_wheel'].float()

        # Fix NaN v odometrii (nahradíme nulou)
        v_left = torch.nan_to_num(odo[:, 0], nan=0.0)
        v_right = torch.nan_to_num(odo[:, 1], nan=0.0)

        # 2. Sestavení vstupu u = [v_l, v_r, theta_imu, omega_imu] (4D)
        # Toto odpovídá "State Model" definovanému v článku (sekce II.C.2)
        u = torch.stack((v_left, v_right, theta_imu, omega_imu), dim=1)

        # 3. Sestavení cíle x (6D) pro state vector [px, py, vx, vy, theta,␣
↪omega]
        # Vyplníme to, co máme z Ground Truth (px, py, theta).
        # Rychlosti (vx, vy, omega) v GT implicitně nemáme (nebo je složité je␣
↪derivovat přesně),
        # ale pro trénink Loss funkce budeme stejně porovnávat primárně pozici.
        T = gt.shape[0]
        x_target = torch.zeros(T, 6)
        x_target[:, 0] = gt[:, 0] # px
        x_target[:, 1] = gt[:, 1] # py
        x_target[:, 4] = gt[:, 2] # theta
        # Ostatní (vx, vy, omega) zůstávají 0, protože v Loss funkci budeme␣
↪maskovat nebo brát jen pozici.

        # 4. Sliding Window (Rozsekání na sekvence)
        num_samples = gt.shape[0]
        current_stride = stride if mode == 'train' else seq_len # U testu bez␣
↪překryvu

        for i in range(0, num_samples - seq_len + 1, current_stride):
            # Cíl: 6D stav
            x_seq = x_target[i : i+seq_len, :]

            # Měření: GPS [px, py]
            y_seq = gps[i : i+seq_len, :]

            # Vstup: 4D control input
```

```python
            u_seq = u[i : i+seq_len, :]

            X_seq_list.append(x_seq)
            Y_seq_list.append(y_seq)
            U_seq_list.append(u_seq)

    # Stack do tenzorů
    X_out = torch.stack(X_seq_list)
    Y_out = torch.stack(Y_seq_list)
    U_out = torch.stack(U_seq_list)

    return X_out, Y_out, U_out

# === NAČTENÍ DAT ===
# Ujistíme se, že cesty a konstanty jsou definované (pokud nejsou, doplňte je␣
↪nahoře)
# if 'DATA_PATH' not in locals(): DATA_PATH = 'data/processed'
# if 'TRAIN_SEQ_LEN' not in locals(): TRAIN_SEQ_LEN = 100
# if 'VAL_SEQ_LEN' not in locals(): VAL_SEQ_LEN = 200
# if 'TEST_SEQ_LEN' not in locals(): TEST_SEQ_LEN = 500
# if 'STRIDE' not in locals(): STRIDE = 20
# if 'BATCH_SIZE' not in locals(): BATCH_SIZE = 256

train_data_raw = torch.load(os.path.join(DATA_PATH, 'train.pt'))
val_data_raw = torch.load(os.path.join(DATA_PATH, 'val.pt'))
test_data_raw = torch.load(os.path.join(DATA_PATH, 'test.pt'))
# Načtení celého balíku
# train_data_full = torch.load(os.path.join(DATA_PATH, 'train.pt'))

# === RYCHLÝ TEST: OŘÍZNUTÍ DAT ===
# Vezmeme jen prvních 5 trajektorií z 22.
# To radikálně zrychlí jednu epochu a umožní rychle otestovat stabilitu␣
↪hyperparametrů.
# train_data_raw = train_data_full[:10]

print(f"DEBUG: Pro rychlý test používám jen {len(train_data_raw)} trajektorií.")
# ... zbytek kódu beze změny

# === PŘÍPRAVA SEKVENCÍ ===
print("--- Generuji trénovací data (Paper compatible) ---")
train_X, train_Y, train_U = prepare_sequences(train_data_raw, TRAIN_SEQ_LEN,␣
↪STRIDE, 'train')

print("\n--- Generuji validační data ---")
val_X, val_Y, val_U = prepare_sequences(val_data_raw, VAL_SEQ_LEN, VAL_SEQ_LEN,␣
↪'val')
```

```python
print("\n--- Generuji testovací data ---")
test_X, test_Y, test_U = prepare_sequences(test_data_raw, TEST_SEQ_LEN,␣
 ↪TEST_SEQ_LEN, 'test')

# Vytvoření DataLoaderů
train_loader = DataLoader(TensorDataset(train_X, train_Y, train_U),␣
 ↪batch_size=BATCH_SIZE, shuffle=True)
val_loader = DataLoader(TensorDataset(val_X, val_Y, val_U),␣
 ↪batch_size=BATCH_SIZE, shuffle=False)
test_loader = DataLoader(TensorDataset(test_X, test_Y, test_U),␣
 ↪batch_size=BATCH_SIZE, shuffle=False)

print(f"\n  Data připravena.")
print(f"Train batches: {len(train_loader)}")
print(f"Shapes -> X: {train_X.shape} (6D State), U: {train_U.shape} (4D Input),␣
 ↪Y: {train_Y.shape} (2D Meas)")
```

```
DEBUG: Pro rychlý test používám jen 22 trajektorií.
--- Generuji trénovací data (Paper compatible) ---
Zpracovávám 22 trajektorií pro train…

--- Generuji validační data ---
Zpracovávám 2 trajektorií pro val…

--- Generuji testovací data ---
Zpracovávám 3 trajektorií pro test…

  Data připravena.
Train batches: 22
Shapes -> X: torch.Size([5504, 50, 6]) (6D State), U: torch.Size([5504, 50, 4])
(4D Input), Y: torch.Size([5504, 50, 2]) (2D Meas)
```

[5]:
```python
# === INICIALIZACE DYNAMICKÉHO MODELU (System Instance - Paper Version) ===

# 1. Parametry systému podle článku [Song et al., 2024]
# State (6D): [px, py, vx, vy, theta, omega]
# Referenční rovnice (5) v článku.
state_dim = 6
# Meas (2D):  [gps_x, gps_y]
# Referenční rovnice (6) v článku.
obs_dim = 2
# Časový krok (z preprocessingu)
dt = 1.0

# 2. Definice Matice Q (Procesní šum / Model Uncertainty)
# Nyní máme 6 stavů. Musíme definovat nejistotu pro každý z nich.
# Hodnoty jsou nastaveny heuristicky (lze ladit):
```

```
# - Pozice (idx 0,1): 0.1
# - Rychlost (idx 2,3): 0.1
# - Úhel/Omega (idx 4,5): 0.01 (IMU je v NCLT docela přesné, ale driftuje)
q_diag = torch.tensor([0.1, 0.1, 0.1, 0.1, 0.01, 0.01])
Q = torch.diag(q_diag)

# 3. Definice Matice R (Šum měření / Sensor Noise)
# GPS měří jen pozici (px, py).
# Nastavujeme 1.0 m^2. To odpovídá standardní odchylce 1m.
# Pokud je GPS v datasetu horší, KalmanNet se naučí "nedůvěřovat" vstupu y
# a spoléhat více na predikci z u (odometrie).
r_diag = torch.tensor([1.0, 1.0])
R = torch.diag(r_diag)

# 4. Počáteční podmínky (Prior)
# Ex0: Nulový vektor 6x1
Ex0 = torch.zeros(state_dim, 1)

# P0: Počáteční kovariance
# Autoři používají P k inicializaci EKF[cite: 700].
# Nastavíme rozumnou počáteční nejistotu.
P0 = torch.eye(state_dim) * 0.5

# 5. Vytvoření instance DynamicSystemNCLT
# Důležité: f=None zajistí, že se použije interní `_f_paper_dynamics` (rovnice␣
↪5),
# která očekává 4D vstup (v_l, v_r, theta, omega).
sys_model = Systems.DynamicSystemNCLT(
    state_dim=state_dim,
    obs_dim=obs_dim,
    Q=Q,
    R=R,
    Ex0=Ex0,
    P0=P0,
    dt=dt,
    f=None,  # None -> Použije se model z článku: px += vc*cos(theta_imu)...
    h=None,  # None -> Použije se GPS model: y = [px, py]
    device=DEVICE
)

print(f" System Model NCLT inicializován (Paper Version).")
print(f" - State Dim: {sys_model.state_dim} [px, py, vx, vy, theta, omega]")
print(f" - Meas Dim:  {sys_model.obs_dim} [gps_x, gps_y]")
print(f" - Input Dim: 4 [v_l, v_r, theta_imu, omega_imu]") # Implicitně v modelu
print(f" - Q Diag: {q_diag.tolist()}")
```

```
System Model NCLT inicializován (Paper Version).
```

```
- State Dim: 6 [px, py, vx, vy, theta, omega]
- Meas Dim:  2 [gps_x, gps_y]
- Input Dim: 4 [v_l, v_r, theta_imu, omega_imu]
- Q Diag: [0.10000000149011612, 0.10000000149011612, 0.10000000149011612,
0.10000000149011612, 0.009999999776482582, 0.009999999776482582]
```

[6]:

```
import torch
import torch.optim as optim
import os
from state_NN_models import StateBayesianKalmanNet
from utils import trainer

state_knet = StateBayesianKalmanNet(
    system_model=sys_model,
    device=DEVICE,
    hidden_size_multiplier=12,       # Větší kapacita pro složitější dynamiku
    output_layer_multiplier=4,
    num_gru_layers=1,                # 1 vrstva GRU obvykle stačí a je␣
↪stabilnější
    init_min_dropout=0.2,
    init_max_dropout=0.4
).to(DEVICE)
print(state_knet)

# Počet trénovatelných parametrů
params_count = sum(p.numel() for p in state_knet.parameters() if p.
↪requires_grad)
print(f"Model má {params_count} trénovatelných parametrů.")
# === 3. PŘÍPRAVA A SPUŠTĚNÍ TRÉNINKU ===
print("\n Spouštím tréninkovou smyčku (Vektorizovanou)...")

# 1. Zjistíme velikost datasetu (počet batchů v jedné epoše)
num_train_batches = len(train_loader)
print(f"INFO: 1 Epocha = {num_train_batches} iterací (batchů).")

# 2. Nastavení parametrů
EPOCHS = 100
WARMUP_EPOCHS = 15  # Prvních 15 epoch jen MSE, aby se model stabilizoval

# Přepočet na iterace (protože nová funkce počítá v iteracích)
total_iterations = EPOCHS * num_train_batches
total_iterations = 1200
warmup_iterations = WARMUP_EPOCHS * num_train_batches
warmup_iterations=350
validation_period = 8  # Chceme validovat jednou za epochu (na konci)

# 3. Volání funkce
```

```python
trained_knet = trainer.train_BayesianKalmanNet(
    model=state_knet,
    train_loader=train_loader,
    val_loader=val_loader,
    device=DEVICE,

    # Parametry iterací
    total_train_iter=total_iterations,
    learning_rate=2e-4,       # Opatrný LR pro BKN
    clip_grad=1.0,            # Stabilizace gradientů
    J_samples=10,              # Počet MC vzorků (paralelně)

    # Logování a validace
    validation_period=validation_period,
    logging_period=10,        # Výpis loss každých 10 batchů

    # Warmup
    warmup_iterations=warmup_iterations,
    weight_decay_=1e-5
)
```

```
INFO: Aplikuji 'Start Zero' inicializaci pro Kalman Gain.
DEBUG: Výstupní vrstva vynulována (Soft Start).
StateBayesianKalmanNet(
  (dnn): DNN_BayesianKalmanNet(
    (input_layer): Sequential(
      (0): Linear(in_features=16, out_features=768, bias=True)
      (1): ReLU()
    )
    (concrete_dropout1): ConcreteDropout()
    (gru): GRU(768, 160)
    (output_layer): Sequential(
      (0): Linear(in_features=160, out_features=48, bias=True)
      (1): ReLU()
      (2): Linear(in_features=48, out_features=12, bias=True)
    )
    (concrete_dropout2): ConcreteDropout()
  )
)
Model má 467774 trénovatelných parametrů.

 Spouštím tréninkovou smyčku (Vektorizovanou)…
INFO: 1 Epocha = 22 iterací (batchů).

[DIAGNOSTIKA iter 0]
  -> Rozdíl (Pred - Target): 1.448191
  -> JE TO PŘESNÁ NULA? Ne
```

```
  -> Min Variance v batchi:  2.85e-03

--- Validation at iteration 8 ---
  Average MSE: 18.3826, Average ANEES: 305.6398
  >>> New best VALIDATION ANEES! Saving model. <<<
--------------------------------------------------
--- Iteration [10/1200] ---
    - Total Loss: 8.9864
    - NLL: 0.0000
    - Reg: 0.1076
    - p1=0.297, p2=0.289

[DIAGNOSTIKA iter 10]
  -> Rozdíl (Pred - Target): 1.050693
  -> JE TO PŘESNÁ NULA? Ne
  -> Min Variance v batchi:  2.69e-03

--- Validation at iteration 16 ---
  Average MSE: 10.7185, Average ANEES: 205.8456
  >>> New best VALIDATION ANEES! Saving model. <<<
--------------------------------------------------
--- Iteration [20/1200] ---
    - Total Loss: 3.3463
    - NLL: 0.0000
    - Reg: 0.1074
    - p1=0.296, p2=0.288

[DIAGNOSTIKA iter 20]
  -> Rozdíl (Pred - Target): 1.008958
  -> JE TO PŘESNÁ NULA? Ne
  -> Min Variance v batchi:  2.35e-03

--- Validation at iteration 24 ---
  Average MSE: 10.2777, Average ANEES: 221.9488
--------------------------------------------------
--- Iteration [30/1200] ---
    - Total Loss: 3.9091
    - NLL: 0.0000
    - Reg: 0.1072
    - p1=0.296, p2=0.288

[DIAGNOSTIKA iter 30]
  -> Rozdíl (Pred - Target): 0.926072
  -> JE TO PŘESNÁ NULA? Ne
  -> Min Variance v batchi:  3.55e-03

--- Validation at iteration 32 ---
  Average MSE: 8.8695, Average ANEES: 182.7519
```

```
  >>> New best VALIDATION ANEES! Saving model. <<<
--------------------------------------------------
--- Iteration [40/1200] ---
    - Total Loss: 2.4382
    - NLL: 0.0000
    - Reg: 0.1071
    - p1=0.295, p2=0.288

--- Validation at iteration 40 ---
  Average MSE: 7.6623, Average ANEES: 170.2187
  >>> New best VALIDATION ANEES! Saving model. <<<
--------------------------------------------------

[DIAGNOSTIKA iter 40]
  -> Rozdíl (Pred - Target): 0.905148
  -> JE TO PŘESNÁ NULA? Ne
  -> Min Variance v batchi:  1.39e-03

--- Validation at iteration 48 ---
  Average MSE: 8.3224, Average ANEES: 185.6198
--------------------------------------------------
--- Iteration [50/1200] ---
    - Total Loss: 2.4905
    - NLL: 0.0000
    - Reg: 0.1069
    - p1=0.295, p2=0.287

[DIAGNOSTIKA iter 50]
  -> Rozdíl (Pred - Target): 0.862936
  -> JE TO PŘESNÁ NULA? Ne
  -> Min Variance v batchi:  6.52e-04

--- Validation at iteration 56 ---
  Average MSE: 12.7638, Average ANEES: 169.9197
  >>> New best VALIDATION ANEES! Saving model. <<<
--------------------------------------------------
--- Iteration [60/1200] ---
    - Total Loss: 3.2413
    - NLL: 0.0000
    - Reg: 0.1068
    - p1=0.295, p2=0.287

[DIAGNOSTIKA iter 60]
  -> Rozdíl (Pred - Target): 0.830302
  -> JE TO PŘESNÁ NULA? Ne
  -> Min Variance v batchi:  3.10e-03

--- Validation at iteration 64 ---
```

```
  Average MSE: 10.5668, Average ANEES: 149.5774
  >>> New best VALIDATION ANEES! Saving model. <<<
--------------------------------------------------
--- Iteration [70/1200] ---
    - Total Loss: 2.3715
    - NLL: 0.0000
    - Reg: 0.1067
    - p1=0.295, p2=0.287

[DIAGNOSTIKA iter 70]
  -> Rozdíl (Pred - Target): 0.809790
  -> JE TO PŘESNÁ NULA? Ne
  -> Min Variance v batchi:  2.96e-03

--- Validation at iteration 72 ---
  Average MSE: 8.1813, Average ANEES: 166.3310
--------------------------------------------------
--- Iteration [80/1200] ---
    - Total Loss: 2.4608
    - NLL: 0.0000
    - Reg: 0.1066
    - p1=0.295, p2=0.287

--- Validation at iteration 80 ---
  Average MSE: 8.2697, Average ANEES: 145.6386
  >>> New best VALIDATION ANEES! Saving model. <<<
--------------------------------------------------

[DIAGNOSTIKA iter 80]
  -> Rozdíl (Pred - Target): 0.857244
  -> JE TO PŘESNÁ NULA? Ne
  -> Min Variance v batchi:  3.58e-03

--- Validation at iteration 88 ---
  Average MSE: 29.2863, Average ANEES: 125.4515
  >>> New best VALIDATION ANEES! Saving model. <<<
--------------------------------------------------
--- Iteration [90/1200] ---
    - Total Loss: 2.6541
    - NLL: 0.0000
    - Reg: 0.1065
    - p1=0.295, p2=0.287

[DIAGNOSTIKA iter 90]
  -> Rozdíl (Pred - Target): 0.884863
  -> JE TO PŘESNÁ NULA? Ne
  -> Min Variance v batchi:  1.53e-03
```

```
--- Validation at iteration 96 ---
  Average MSE: 9.2138, Average ANEES: 143.5436
--------------------------------------------------
--- Iteration [100/1200] ---
    - Total Loss: 2.2721
    - NLL: 0.0000
    - Reg: 0.1065
    - p1=0.295, p2=0.287

[DIAGNOSTIKA iter 100]
  -> Rozdíl (Pred - Target): 0.851493
  -> JE TO PŘESNÁ NULA? Ne
  -> Min Variance v batchi:  2.91e-03

--- Validation at iteration 104 ---
  Average MSE: 8.6723, Average ANEES: 137.2831
--------------------------------------------------
--- Iteration [110/1200] ---
    - Total Loss: 1.8750
    - NLL: 0.0000
    - Reg: 0.1064
    - p1=0.295, p2=0.287

[DIAGNOSTIKA iter 110]
  -> Rozdíl (Pred - Target): 0.772532
  -> JE TO PŘESNÁ NULA? Ne
  -> Min Variance v batchi:  1.74e-03

--- Validation at iteration 112 ---
  Average MSE: 10.8951, Average ANEES: 144.3659
--------------------------------------------------
--- Iteration [120/1200] ---
    - Total Loss: 2.3509
    - NLL: 0.0000
    - Reg: 0.1063
    - p1=0.294, p2=0.287

--- Validation at iteration 120 ---
  Average MSE: 11.2595, Average ANEES: 128.5785
--------------------------------------------------

[DIAGNOSTIKA iter 120]
  -> Rozdíl (Pred - Target): 0.772150
  -> JE TO PŘESNÁ NULA? Ne
  -> Min Variance v batchi:  3.07e-03

--- Validation at iteration 128 ---
  Average MSE: 6.5807, Average ANEES: 109.6478
```

```
  >>> New best VALIDATION ANEES! Saving model. <<<
--------------------------------------------------
--- Iteration [130/1200] ---
    - Total Loss: 2.2881
    - NLL: 0.0000
    - Reg: 0.1062
    - p1=0.294, p2=0.286

[DIAGNOSTIKA iter 130]
  -> Rozdíl (Pred - Target): 0.758511
  -> JE TO PŘESNÁ NULA? Ne
  -> Min Variance v batchi:  2.61e-03

--- Validation at iteration 136 ---
  Average MSE: 7.8598, Average ANEES: 108.7271
  >>> New best VALIDATION ANEES! Saving model. <<<
--------------------------------------------------
--- Iteration [140/1200] ---
    - Total Loss: 2.5756
    - NLL: 0.0000
    - Reg: 0.1062
    - p1=0.294, p2=0.286

[DIAGNOSTIKA iter 140]
  -> Rozdíl (Pred - Target): 0.784809
  -> JE TO PŘESNÁ NULA? Ne
  -> Min Variance v batchi:  2.90e-03

--- Validation at iteration 144 ---
  Average MSE: 8.7422, Average ANEES: 128.1277
--------------------------------------------------
--- Iteration [150/1200] ---
    - Total Loss: 1.9082
    - NLL: 0.0000
    - Reg: 0.1061
    - p1=0.294, p2=0.286

[DIAGNOSTIKA iter 150]
  -> Rozdíl (Pred - Target): 0.744708
  -> JE TO PŘESNÁ NULA? Ne
  -> Min Variance v batchi:  2.37e-03

--- Validation at iteration 152 ---
  Average MSE: 8.2816, Average ANEES: 98.4389
  >>> New best VALIDATION ANEES! Saving model. <<<
--------------------------------------------------
--- Iteration [160/1200] ---
    - Total Loss: 3.1972
```

```
    - NLL: 0.0000
    - Reg: 0.1061
    - p1=0.294, p2=0.286

--- Validation at iteration 160 ---
  Average MSE: 9.5339, Average ANEES: 112.2533
--------------------------------------------------

[DIAGNOSTIKA iter 160]
  -> Rozdíl (Pred - Target): 0.736997
  -> JE TO PŘESNÁ NULA? Ne
  -> Min Variance v batchi:  2.18e-03

--- Validation at iteration 168 ---
  Average MSE: 8.6311, Average ANEES: 101.0700
--------------------------------------------------
--- Iteration [170/1200] ---
    - Total Loss: 2.5270
    - NLL: 0.0000
    - Reg: 0.1060
    - p1=0.294, p2=0.286

[DIAGNOSTIKA iter 170]
  -> Rozdíl (Pred - Target): 0.753578
  -> JE TO PŘESNÁ NULA? Ne
  -> Min Variance v batchi:  2.90e-03

--- Validation at iteration 176 ---
  Average MSE: 9.8109, Average ANEES: 110.6072
--------------------------------------------------
--- Iteration [180/1200] ---
    - Total Loss: 2.3562
    - NLL: 0.0000
    - Reg: 0.1060
    - p1=0.294, p2=0.286

[DIAGNOSTIKA iter 180]
  -> Rozdíl (Pred - Target): 0.770882
  -> JE TO PŘESNÁ NULA? Ne
  -> Min Variance v batchi:  2.73e-03

--- Validation at iteration 184 ---
  Average MSE: 6.9270, Average ANEES: 124.7723
--------------------------------------------------
--- Iteration [190/1200] ---
    - Total Loss: 2.1732
    - NLL: 0.0000
    - Reg: 0.1059
```

```
    - p1=0.294, p2=0.286

[DIAGNOSTIKA iter 190]
  -> Rozdíl (Pred - Target): 0.718990
  -> JE TO PŘESNÁ NULA? Ne
  -> Min Variance v batchi:  1.82e-03

--- Validation at iteration 192 ---
  Average MSE: 7.8692, Average ANEES: 101.1646
--------------------------------------------------
--- Iteration [200/1200] ---
    - Total Loss: 2.1227
    - NLL: 0.0000
    - Reg: 0.1059
    - p1=0.294, p2=0.285

--- Validation at iteration 200 ---
  Average MSE: 8.1890, Average ANEES: 107.7434
--------------------------------------------------

[DIAGNOSTIKA iter 200]
  -> Rozdíl (Pred - Target): 0.758725
  -> JE TO PŘESNÁ NULA? Ne
  -> Min Variance v batchi:  1.85e-03

--- Validation at iteration 208 ---
  Average MSE: 7.5328, Average ANEES: 133.0555
--------------------------------------------------
--- Iteration [210/1200] ---
    - Total Loss: 2.2956
    - NLL: 0.0000
    - Reg: 0.1058
    - p1=0.294, p2=0.285

[DIAGNOSTIKA iter 210]
  -> Rozdíl (Pred - Target): 0.766259
  -> JE TO PŘESNÁ NULA? Ne
  -> Min Variance v batchi:  2.85e-03

--- Validation at iteration 216 ---
  Average MSE: 11.2123, Average ANEES: 127.4420
--------------------------------------------------
--- Iteration [220/1200] ---
    - Total Loss: 4.6965
    - NLL: 0.0000
    - Reg: 0.1057
    - p1=0.294, p2=0.285
```

```
[DIAGNOSTIKA iter 220]
  -> Rozdíl (Pred - Target): 0.755984
  -> JE TO PŘESNÁ NULA? Ne
  -> Min Variance v batchi:  2.45e-03

--- Validation at iteration 224 ---
  Average MSE: 9.1819, Average ANEES: 106.9758
--------------------------------------------------
--- Iteration [230/1200] ---
    - Total Loss: 2.0153
    - NLL: 0.0000
    - Reg: 0.1057
    - p1=0.294, p2=0.285

[DIAGNOSTIKA iter 230]
  -> Rozdíl (Pred - Target): 0.723199
  -> JE TO PŘESNÁ NULA? Ne
  -> Min Variance v batchi:  2.85e-03

--- Validation at iteration 232 ---
  Average MSE: 10.0161, Average ANEES: 91.4551
  >>> New best VALIDATION ANEES! Saving model. <<<
--------------------------------------------------
--- Iteration [240/1200] ---
    - Total Loss: 2.3899
    - NLL: 0.0000
    - Reg: 0.1056
    - p1=0.293, p2=0.285

--- Validation at iteration 240 ---
  Average MSE: 9.5528, Average ANEES: 101.8019
--------------------------------------------------

[DIAGNOSTIKA iter 240]
  -> Rozdíl (Pred - Target): 0.735398
  -> JE TO PŘESNÁ NULA? Ne
  -> Min Variance v batchi:  2.65e-03

--- Validation at iteration 248 ---
  Average MSE: 7.2535, Average ANEES: 131.3935
--------------------------------------------------
--- Iteration [250/1200] ---
    - Total Loss: 2.9157
    - NLL: 0.0000
    - Reg: 0.1055
    - p1=0.293, p2=0.285

[DIAGNOSTIKA iter 250]
```

```
  -> Rozdíl (Pred - Target): 0.750718
  -> JE TO PŘESNÁ NULA? Ne
  -> Min Variance v batchi:  3.39e-03

--- Validation at iteration 256 ---
  Average MSE: 11.2802, Average ANEES: 108.0070
--------------------------------------------------
--- Iteration [260/1200] ---
    - Total Loss: 1.9214
    - NLL: 0.0000
    - Reg: 0.1055
    - p1=0.293, p2=0.285

[DIAGNOSTIKA iter 260]
  -> Rozdíl (Pred - Target): 0.744866
  -> JE TO PŘESNÁ NULA? Ne
  -> Min Variance v batchi:  2.14e-03

--- Validation at iteration 264 ---
  Average MSE: 8.6124, Average ANEES: 91.2030
  >>> New best VALIDATION ANEES! Saving model. <<<
--------------------------------------------------
--- Iteration [270/1200] ---
    - Total Loss: 2.6596
    - NLL: 0.0000
    - Reg: 0.1054
    - p1=0.293, p2=0.285

[DIAGNOSTIKA iter 270]
  -> Rozdíl (Pred - Target): 0.712032
  -> JE TO PŘESNÁ NULA? Ne
  -> Min Variance v batchi:  3.25e-03

--- Validation at iteration 272 ---
  Average MSE: 6.5890, Average ANEES: 89.7190
  >>> New best VALIDATION ANEES! Saving model. <<<
--------------------------------------------------
--- Iteration [280/1200] ---
    - Total Loss: 2.7497
    - NLL: 0.0000
    - Reg: 0.1053
    - p1=0.293, p2=0.285

--- Validation at iteration 280 ---
  Average MSE: 6.8707, Average ANEES: 120.1964
--------------------------------------------------

[DIAGNOSTIKA iter 280]
```

```
  -> Rozdíl (Pred - Target): 0.693749
  -> JE TO PŘESNÁ NULA? Ne
  -> Min Variance v batchi:  9.82e-04

--- Validation at iteration 288 ---
  Average MSE: 8.9645, Average ANEES: 88.1623
  >>> New best VALIDATION ANEES! Saving model. <<<
--------------------------------------------------
--- Iteration [290/1200] ---
    - Total Loss: 3.0205
    - NLL: 0.0000
    - Reg: 0.1053
    - p1=0.293, p2=0.284

[DIAGNOSTIKA iter 290]
  -> Rozdíl (Pred - Target): 0.687245
  -> JE TO PŘESNÁ NULA? Ne
  -> Min Variance v batchi:  2.64e-03

--- Validation at iteration 296 ---
  Average MSE: 9.1173, Average ANEES: 95.6676
--------------------------------------------------
--- Iteration [300/1200] ---
    - Total Loss: 2.1458
    - NLL: 0.0000
    - Reg: 0.1053
    - p1=0.293, p2=0.284

[DIAGNOSTIKA iter 300]
  -> Rozdíl (Pred - Target): 0.766157
  -> JE TO PŘESNÁ NULA? Ne
  -> Min Variance v batchi:  3.31e-03

--- Validation at iteration 304 ---
  Average MSE: 7.8276, Average ANEES: 101.9332
--------------------------------------------------
--- Iteration [310/1200] ---
    - Total Loss: 2.2848
    - NLL: 0.0000
    - Reg: 0.1052
    - p1=0.293, p2=0.284

[DIAGNOSTIKA iter 310]
  -> Rozdíl (Pred - Target): 0.754141
  -> JE TO PŘESNÁ NULA? Ne
  -> Min Variance v batchi:  8.15e-04

--- Validation at iteration 312 ---
```

```
  Average MSE: 8.6508, Average ANEES: 116.0737
--------------------------------------------------
--- Iteration [320/1200] ---
    - Total Loss: 1.9395
    - NLL: 0.0000
    - Reg: 0.1052
    - p1=0.293, p2=0.284

--- Validation at iteration 320 ---
  Average MSE: 7.4918, Average ANEES: 96.5788
--------------------------------------------------

[DIAGNOSTIKA iter 320]
  -> Rozdíl (Pred - Target): 0.768997
  -> JE TO PŘESNÁ NULA? Ne
  -> Min Variance v batchi:  2.52e-03

--- Validation at iteration 328 ---
  Average MSE: 6.9817, Average ANEES: 89.5381
--------------------------------------------------
--- Iteration [330/1200] ---
    - Total Loss: 2.8597
    - NLL: 0.0000
    - Reg: 0.1052
    - p1=0.293, p2=0.284

[DIAGNOSTIKA iter 330]
  -> Rozdíl (Pred - Target): 0.763441
  -> JE TO PŘESNÁ NULA? Ne
  -> Min Variance v batchi:  2.95e-03

--- Validation at iteration 336 ---
  Average MSE: 7.6424, Average ANEES: 87.4958
  >>> New best VALIDATION ANEES! Saving model. <<<
--------------------------------------------------
--- Iteration [340/1200] ---
    - Total Loss: 2.4047
    - NLL: 0.0000
    - Reg: 0.1052
    - p1=0.293, p2=0.284

[DIAGNOSTIKA iter 340]
  -> Rozdíl (Pred - Target): 0.744171
  -> JE TO PŘESNÁ NULA? Ne
  -> Min Variance v batchi:  1.52e-03

--- Validation at iteration 344 ---
  Average MSE: 8.1121, Average ANEES: 95.5952
```

```
--------------------------------------------------
--- Iteration [350/1200] ---
    - Total Loss: 2.2574
    - NLL: 0.0000
    - Reg: 0.1052
    - p1=0.293, p2=0.284

[DIAGNOSTIKA iter 350]
  -> Rozdíl (Pred - Target): 0.702179
  -> JE TO PŘESNÁ NULA? Ne
  -> Min Variance v batchi:  3.06e-03

--- Validation at iteration 352 ---
  Average MSE: 8.3490, Average ANEES: 88.6104
--------------------------------------------------
--- Iteration [360/1200] ---
    - Total Loss: 0.5302
    - NLL: 0.4251
    - Reg: 0.1051
    - p1=0.293, p2=0.284

--- Validation at iteration 360 ---
  Average MSE: 9.9811, Average ANEES: 73.9907
  >>> New best VALIDATION ANEES! Saving model. <<<
--------------------------------------------------

[DIAGNOSTIKA iter 360]
  -> Rozdíl (Pred - Target): 0.774273
  -> JE TO PŘESNÁ NULA? Ne
  -> Min Variance v batchi:  2.81e-03

--- Validation at iteration 368 ---
  Average MSE: 10.1048, Average ANEES: 67.6924
  >>> New best VALIDATION ANEES! Saving model. <<<
--------------------------------------------------
--- Iteration [370/1200] ---
    - Total Loss: 0.3877
    - NLL: 0.2829
    - Reg: 0.1048
    - p1=0.293, p2=0.283

[DIAGNOSTIKA iter 370]
  -> Rozdíl (Pred - Target): 0.759004
  -> JE TO PŘESNÁ NULA? Ne
  -> Min Variance v batchi:  2.35e-03

--- Validation at iteration 376 ---
  Average MSE: 8.4918, Average ANEES: 54.4096
```

```
  >>> New best VALIDATION ANEES! Saving model. <<<
--------------------------------------------------
--- Iteration [380/1200] ---
    - Total Loss: 0.4389
    - NLL: 0.3345
    - Reg: 0.1044
    - p1=0.292, p2=0.282

[DIAGNOSTIKA iter 380]
  -> Rozdíl (Pred - Target): 0.732184
  -> JE TO PŘESNÁ NULA? Ne
  -> Min Variance v batchi:  2.39e-03

--- Validation at iteration 384 ---
  Average MSE: 9.0617, Average ANEES: 59.8846
--------------------------------------------------
--- Iteration [390/1200] ---
    - Total Loss: 0.3358
    - NLL: 0.2318
    - Reg: 0.1040
    - p1=0.292, p2=0.281

[DIAGNOSTIKA iter 390]
  -> Rozdíl (Pred - Target): 0.716510
  -> JE TO PŘESNÁ NULA? Ne
  -> Min Variance v batchi:  2.91e-03

--- Validation at iteration 392 ---
  Average MSE: 7.2162, Average ANEES: 50.5501
  >>> New best VALIDATION ANEES! Saving model. <<<
--------------------------------------------------
--- Iteration [400/1200] ---
    - Total Loss: 0.4943
    - NLL: 0.3908
    - Reg: 0.1036
    - p1=0.291, p2=0.281

--- Validation at iteration 400 ---
  Average MSE: 7.6906, Average ANEES: 47.8508
  >>> New best VALIDATION ANEES! Saving model. <<<
--------------------------------------------------

[DIAGNOSTIKA iter 400]
  -> Rozdíl (Pred - Target): 0.711236
  -> JE TO PŘESNÁ NULA? Ne
  -> Min Variance v batchi:  2.32e-03

--- Validation at iteration 408 ---
```

```
  Average MSE: 8.9384, Average ANEES: 49.7112
--------------------------------------------------
--- Iteration [410/1200] ---
    - Total Loss: 0.3067
    - NLL: 0.2036
    - Reg: 0.1031
    - p1=0.291, p2=0.280

[DIAGNOSTIKA iter 410]
  -> Rozdíl (Pred - Target): 0.735218
  -> JE TO PŘESNÁ NULA? Ne
  -> Min Variance v batchi:  2.59e-03

--- Validation at iteration 416 ---
  Average MSE: 7.0623, Average ANEES: 44.6161
  >>> New best VALIDATION ANEES! Saving model. <<<
--------------------------------------------------
--- Iteration [420/1200] ---
    - Total Loss: 0.3235
    - NLL: 0.2208
    - Reg: 0.1027
    - p1=0.291, p2=0.279

[DIAGNOSTIKA iter 420]
  -> Rozdíl (Pred - Target): 0.685599
  -> JE TO PŘESNÁ NULA? Ne
  -> Min Variance v batchi:  1.63e-03

--- Validation at iteration 424 ---
  Average MSE: 7.6984, Average ANEES: 56.2356
--------------------------------------------------
--- Iteration [430/1200] ---
    - Total Loss: 0.3106
    - NLL: 0.2084
    - Reg: 0.1022
    - p1=0.290, p2=0.278

[DIAGNOSTIKA iter 430]
  -> Rozdíl (Pred - Target): 0.741956
  -> JE TO PŘESNÁ NULA? Ne
  -> Min Variance v batchi:  2.29e-03

--- Validation at iteration 432 ---
  Average MSE: 7.8570, Average ANEES: 51.6483
--------------------------------------------------
--- Iteration [440/1200] ---
    - Total Loss: 0.4033
    - NLL: 0.3015
```

```
    - Reg: 0.1018
    - p1=0.290, p2=0.277

--- Validation at iteration 440 ---
  Average MSE: 7.0812, Average ANEES: 47.7836
--------------------------------------------------

[DIAGNOSTIKA iter 440]
  -> Rozdíl (Pred - Target): 0.699231
  -> JE TO PŘESNÁ NULA? Ne
  -> Min Variance v batchi:  1.02e-03

--- Validation at iteration 448 ---
  Average MSE: 8.8849, Average ANEES: 46.8978
--------------------------------------------------
--- Iteration [450/1200] ---
    - Total Loss: 0.3076
    - NLL: 0.2062
    - Reg: 0.1014
    - p1=0.289, p2=0.276

[DIAGNOSTIKA iter 450]
  -> Rozdíl (Pred - Target): 0.734303
  -> JE TO PŘESNÁ NULA? Ne
  -> Min Variance v batchi:  2.35e-03

--- Validation at iteration 456 ---
  Average MSE: 7.8507, Average ANEES: 52.9697
--------------------------------------------------
--- Iteration [460/1200] ---
    - Total Loss: 0.3271
    - NLL: 0.2261
    - Reg: 0.1009
    - p1=0.289, p2=0.276

[DIAGNOSTIKA iter 460]
  -> Rozdíl (Pred - Target): 0.709238
  -> JE TO PŘESNÁ NULA? Ne
  -> Min Variance v batchi:  2.74e-03

--- Validation at iteration 464 ---
  Average MSE: 6.7832, Average ANEES: 51.3750
--------------------------------------------------
--- Iteration [470/1200] ---
    - Total Loss: 0.3264
    - NLL: 0.2259
    - Reg: 0.1005
    - p1=0.288, p2=0.275
```

```
[DIAGNOSTIKA iter 470]
  -> Rozdíl (Pred - Target): 0.686134
  -> JE TO PŘESNÁ NULA? Ne
  -> Min Variance v batchi:  2.35e-03

--- Validation at iteration 472 ---
  Average MSE: 8.0103, Average ANEES: 45.2692
--------------------------------------------------
--- Iteration [480/1200] ---
    - Total Loss: 0.2627
    - NLL: 0.1626
    - Reg: 0.1001
    - p1=0.288, p2=0.274

--- Validation at iteration 480 ---
  Average MSE: 11.1989, Average ANEES: 47.6736
--------------------------------------------------

[DIAGNOSTIKA iter 480]
  -> Rozdíl (Pred - Target): 0.672207
  -> JE TO PŘESNÁ NULA? Ne
  -> Min Variance v batchi:  3.32e-03
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
Cell In[6], line 40
     37 validation_period = 8  # Chceme validovat jednou za epochu (na konci)
     39 # 3. Volání funkce
---> 40 trained_knet = trainer.train_BayesianKalmanNet(
     41     model=state_knet,
     42     train_loader=train_loader,
     43     val_loader=val_loader,
     44     device=DEVICE,
     45 
     46     # Parametry iterací
     47     total_train_iter=total_iterations,
     48     learning_rate=2e-4,       # Opatrný LR pro BKN
     49     clip_grad=1.0,            # Stabilizace gradientů
     50     J_samples=10,              # Počet MC vzorků (paralelně)
     51 
     52     # Logování a validace
     53     validation_period=validation_period,
     54     logging_period=10,        # Výpis loss každých 10 batchů
     55 
     56     # Warmup
     57     warmup_iterations=warmup_iterations,
```

```
     58     weight_decay_=1e-5
     59 )

File ~/skola/KalmanNet-for-state-estimation/utils/trainer.py:1773, in train_BayesianKalmanNet(model, train_loader, val_loader, device, total_train_iter, learning_rate, clip_grad, J_samples, validation_period, logging_period, warmup_iterations, weight_decay_)
   1771 y_t = y_meas_batch[:, t, :]
   1772 u_t = u_input_batch[:,t,:]
-> 1773 x_filtered_t, reg_t = model.step(y_t,u_t)
   1774 if torch.isnan(x_filtered_t).any():
   1775         raise ValueError(f"NaN in x_filtered_t at sample {j}, step {t}"

File ~/skola/KalmanNet-for-state-estimation/state_NN_models/StateBayesianKalmanNet.py:181, in StateBayesianKalmanNet.step(self, y_t, u_t)
    178 self._check_tensor(raw_correction, "Raw Correction (K * Innovation)")
    180 correction = raw_correction * valid_mask.float()
--> 181 correction = torch.clamp(correction, min=-10.0, max=10.0)
    182 # POZOR: Pokud jsi zakomentoval clamp, zde může dojít k explozi
    183 # Doporučuji nechat alespoň měkký clamp pro debug
    184 # correction = torch.clamp(correction, min=-50.0, max=50.0)
    186 x_filtered = x_predicted + correction

KeyboardInterrupt:
```

[ ]:

```python
if False:
    import torch
    import os

    # Protože jsi trénink přerušil, nemáme slovník s metrikami automaticky.
    # Musíme uložit přímo model 'state_knet'.

    # 1. Definuj cestu (Metriky si do názvu musíš dopsat ručně podle toho, co jsi viděl v logu naposledy,
    # nebo použij obecný název, abys o model nepřišel).
    save_path = 'best_BKN_test_results_.pth'

    # Pokud si pamatuješ hodnoty z logu (např. ANEES 23.5), můžeš je tam dopsat ručně:
    # save_path = 'best_kalmannet_nclt_sensor_fusion_ANEES23.57_MSE8.17_interrupted.pth'

    # 2. Uložení
    # DŮLEŽITÉ: Voláme .state_dict() přímo na objektu state_knet (nebo trained_knet, pokud jsi ho přiřadil)
    # Nepoužíváme závorky ['final_model'], protože state_knet UŽ JE ten model.
    torch.save(state_knet.state_dict(), save_path)
```

```
    print(f"\n Záchrana modelu úspěšná! Uloženo do: {save_path}")

if False:
    import os
    trained_knet = state_knet
    # Přístup k metrikám ve slovníku je správně (přes závorky [])
    print(f"   -> Best ANEES: {trained_knet['best_val_anees']:.4f}")
    print(f"   -> Best MSE:   {trained_knet['best_val_mse']:.4f}")
    save_path =␣
 ↪f'best_kalmannet_nclt_sensor_fusion_ANEES{trained_knet["best_val_anees"]:.
 ↪4f}_MSE{trained_knet["best_val_mse"]:.4f}_nejlepsi_test_vysledky.pth '

    # OPRAVA: Musíš vytáhnout model ze slovníku pomocí klíče 'final_model'
    # A teprve na něm zavolat .state_dict()
    torch.save(trained_knet['final_model'].state_dict(), save_path)

    print(f"\n Model úspěšně uložen do: {save_path}")
```

```
Záchrana modelu úspěšná! Uloženo do: best_BKN_test_results.pth
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[10], line 26
     24 trained_knet = state_knet
     25 # Přístup k metrikám ve slovníku je správně (přes závorky [])
---> 26 print(f"   -> Best ANEES: {trained_knet['best_val_anees']:.4f}")
     27 print(f"   -> Best MSE:   {trained_knet['best_val_mse']:.4f}")
     28 save_path =␣
 ↪f'best_kalmannet_nclt_sensor_fusion_ANEES{trained_knet["best_val_anees"]:.
 ↪4f}_MSE{trained_knet["best_val_mse"]:.4f}_nejlepsi_test_vysledky.pth '

TypeError: 'StateBayesianKalmanNet' object is not subscriptable
```

[7]:
```
import torch
import torch.nn.functional as F
import numpy as np
import matplotlib.pyplot as plt
import Filters

# ===============================================================================
# 0. KONFIGURACE A MODEL
# ===============================================================================
DT_SEC = 1.0
J_SAMPLES = 50  # 50 vzorků pro kvalitní odhad distribuce
```

```python
if hasattr(sys_model, 'dt'):
    sys_model.dt = DT_SEC
    print(f"INFO: Nastaveno sys_model.dt = {DT_SEC} s")

# Načtení Bayesian modelu
try:
    trained_model_bkn = state_knet
    trained_model_bkn.eval()
    print(f"INFO: Používám Bayesian KalmanNet s J={J_SAMPLES} vzorky.")
except NameError:
    raise NameError("Chyba: 'state_knet' (BKN) neexistuje.")

# Inicializace klasických filtrů
ukf_filter = Filters.UnscentedKalmanFilter(sys_model)
ekf_filter = Filters.ExtendedKalmanFilter(sys_model)

# --- ROBUSTNÍ FUNKCE PRO ANEES ---
def calculate_anees(x_true, x_est, P_est):
    """
    Vypočítá ANEES. Očekává vstupy už se shodnou dimenzí.
    Používá pinv pro stabilitu.
    """
    T = x_true.shape[0]
    anees_list = []
    error = x_true - x_est

    for t in range(T):
        e_t = error[t].unsqueeze(1) # [n, 1]
        P_t = P_est[t]              # [n, n]

        try:
            P_inv = torch.linalg.pinv(P_t, hermitian=True)
        except RuntimeError:
            P_inv = torch.eye(P_t.shape[0], device=P_t.device)

        nees = torch.mm(torch.mm(e_t.t(), P_inv), e_t).item()
        anees_list.append(nees)

    return np.mean(anees_list)

# ==============================================================================
# 1. EVALUACE A VIZUALIZACE
# ==============================================================================
results = {
    'GPS_Sensor': [],   'GPS_Authors': [],
    'EKF_MSE': [],      'EKF_ANEES': [],
    'UKF_MSE': [],      'UKF_ANEES': [],
```

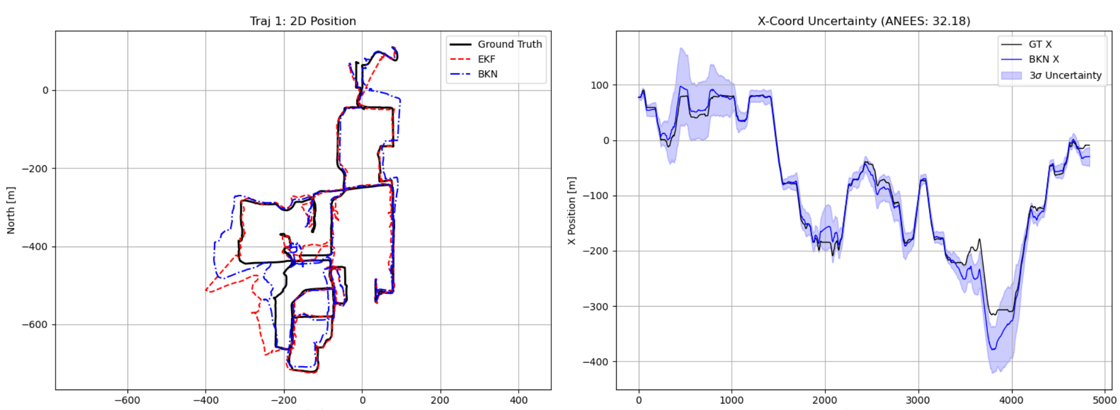
```
    'BKN_MSE': [],      'BKN_ANEES': []
}

print(f"\nSpouštím detailní evaluaci na {len(test_data_raw)} trajektoriích...")

# HLAVIČKA DETAILNÍ TABULKY
print("\n" + "-"*95)
print(f"{'TRAJEKTORIE':<12} | {'BKN MSE':<10} | {'BKN ANEES':<10} | {'EKF MSE':
 ↪<10} | {'EKF ANEES':<10} | {'GPS MSE':<10}")
print("-" * 95)

for i, traj in enumerate(test_data_raw):
    # --- PŘÍPRAVA DAT ---
    gt_raw = traj['ground_truth'].float().to(DEVICE)
    gps_filtered = traj['filtered_gps'].float().to(DEVICE)
    gps_filled = traj['gps'].float().to(DEVICE)
    imu_raw = traj['imu'].float().to(DEVICE)
    odo_raw = traj['filtered_wheel'].float().to(DEVICE)

    T_len = gt_raw.shape[0]

    # Input vector
    u_full = torch.stack((
        torch.nan_to_num(odo_raw[:, 0], nan=0.0),
        torch.nan_to_num(odo_raw[:, 1], nan=0.0),
        imu_raw[:, 2],
        imu_raw[:, 3]
    ), dim=1).to(DEVICE)

    # Init State (bereme z GT)
    # x_true má dimenzi 3 (X, Y, Theta)
    x_true = gt_raw[:, :3]

    # Init pro filtry (plná dimenze modelu)
    m = sys_model.state_dim
    x0_vec = torch.zeros(m).to(DEVICE)
    x0_vec[0] = x_true[0, 0]; x0_vec[1] = x_true[0, 1]
    if m >= 3 and x_true.shape[1] >= 3: x0_vec[4] = x_true[0, 2]

    # --- A. BĚH BAYESIAN KALMANNET ---
    trained_model_bkn.train() # Nutné pro MC Dropout

    batch_x0 = x0_vec.unsqueeze(0).repeat(J_SAMPLES, 1)
    trained_model_bkn.reset(batch_size=J_SAMPLES, initial_state=batch_x0)

    bkn_samples = []
    bkn_samples.append(batch_x0.unsqueeze(1))
```

```
    with torch.no_grad():
        for t in range(1, T_len):
            y_t = gps_filtered[t].unsqueeze(0).repeat(J_SAMPLES, 1)
            u_t = u_full[t].unsqueeze(0).repeat(J_SAMPLES, 1)
            x_est_j, _ = trained_model_bkn.step(y_t, u_t)
            bkn_samples.append(x_est_j.unsqueeze(1))

    trained_model_bkn.eval()
    bkn_ensemble = torch.cat(bkn_samples, dim=1) # [J, T, m]

    # Statistiky BKN
    x_bkn_mean = torch.mean(bkn_ensemble, dim=0)

    # Výpočet kovariance BKN
    centered = bkn_ensemble - x_bkn_mean.unsqueeze(0)
    centered_perm = centered.permute(1, 2, 0)
    P_bkn = torch.bmm(centered_perm, centered_perm.transpose(1, 2)) / 
↪(J_SAMPLES - 1)
    P_bkn = P_bkn + torch.eye(m, device=DEVICE).unsqueeze(0) * 1e-6

    # --- B. EKF & UKF ---
    def run_filter(flt):
        try:
            res = flt.process_sequence(gps_filtered, u_seq=u_full, Ex0=x0_vec, 
↪P0=sys_model.P0)
            return res['x_filtered'], res.get('P_filtered', torch.eye(m, 
↪device=DEVICE).repeat(T_len, 1, 1))
        except:
            return torch.zeros(T_len, m).to(DEVICE), torch.eye(m, 
↪device=DEVICE).repeat(T_len, 1, 1)

    x_ekf, P_ekf = run_filter(ekf_filter)
    x_ukf, P_ukf = run_filter(ukf_filter)

    # --- VYHODNOCENÍ (SLICING) ---
    eval_dim = x_true.shape[1] # 3 (X, Y, Theta)

    def evaluate_metrics(est_x, est_P):
        # MSE (poloha X, Y)
        mse = F.mse_loss(est_x[:, :2], x_true[:, :2]).item()
        # ANEES (Oříznuté na eval_dim)
        est_x_sliced = est_x[:, :eval_dim]
        est_P_sliced = est_P[:, :eval_dim, :eval_dim]
        anees = calculate_anees(x_true, est_x_sliced, est_P_sliced)
        return mse, anees
```

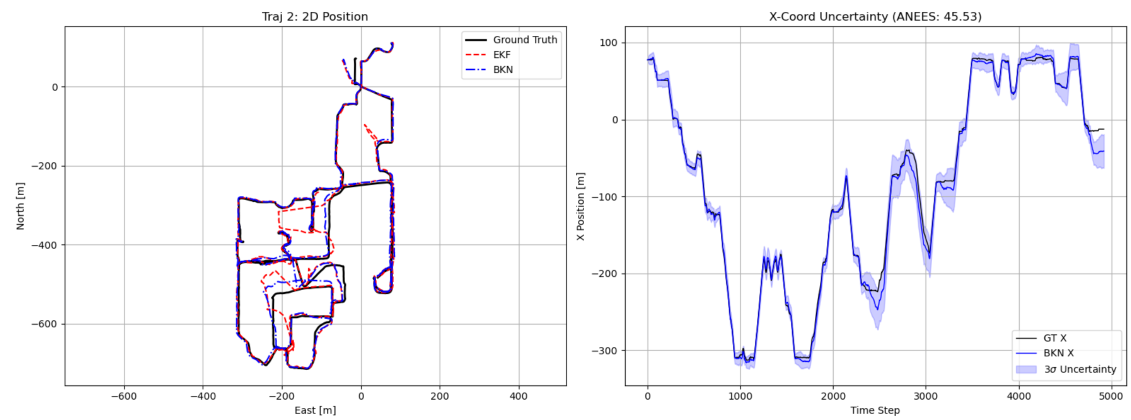
```
    mse_ekf, anees_ekf = evaluate_metrics(x_ekf, P_ekf)
    mse_ukf, anees_ukf = evaluate_metrics(x_ukf, P_ukf)
    mse_bkn, anees_bkn = evaluate_metrics(x_bkn_mean, P_bkn)

    valid_mask = ~torch.isnan(gps_filtered[:, 0])
    mse_gps = F.mse_loss(gps_filtered[valid_mask], x_true[valid_mask, :2]).
↪item() if valid_mask.sum() > 0 else np.nan

    # Ukládání
    results['GPS_Sensor'].append(mse_gps)
    results['EKF_MSE'].append(mse_ekf); results['EKF_ANEES'].append(anees_ekf)
    results['UKF_MSE'].append(mse_ukf); results['UKF_ANEES'].append(anees_ukf)
    results['BKN_MSE'].append(mse_bkn); results['BKN_ANEES'].append(anees_bkn)

    # --- VÝPIS ŘÁDKU TABULKY ---
    print(f"{i+1:<12} | {mse_bkn:<10.2f} | {anees_bkn:<10.2f} | {mse_ekf:<10.
↪2f} | {anees_ekf:<10.2f} | {mse_gps:<10.2f}")

    # --- VIZUALIZACE ---
    gt_np = x_true.cpu().numpy()
    bkn_np = x_bkn_mean.cpu().numpy()
    std_x_bkn = torch.sqrt(P_bkn[:, 0, 0]).cpu().numpy()
    t_axis = np.arange(T_len)

    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(16, 6))

    # Graf 1: 2D Trajektorie
    ax1.plot(gt_np[:, 0], gt_np[:, 1], 'k-', lw=2, label='Ground Truth')
    ax1.plot(x_ekf.cpu().numpy()[:, 0], x_ekf.cpu().numpy()[:, 1], 'r--', lw=1.
↪5, label='EKF')
    ax1.plot(bkn_np[:, 0], bkn_np[:, 1], 'b-.', lw=1.5, label='BKN')
    ax1.set_title(f'Traj {i+1}: 2D Position')
    ax1.set_xlabel('East [m]'); ax1.set_ylabel('North [m]')
    ax1.legend(); ax1.grid(True); ax1.axis('equal')

    # Graf 2: Time Series X s Confidence Tube
    ax2.plot(t_axis, gt_np[:, 0], 'k-', lw=1, label='GT X')
    ax2.plot(t_axis, bkn_np[:, 0], 'b-', lw=1, label='BKN X')
    ax2.fill_between(t_axis,
                     bkn_np[:, 0] - 3*std_x_bkn,
                     bkn_np[:, 0] + 3*std_x_bkn,
                     color='blue', alpha=0.2, label='3$\sigma$ Uncertainty')
    ax2.set_title(f'X-Coord Uncertainty (ANEES: {anees_bkn:.2f})')
    ax2.set_xlabel('Time Step'); ax2.set_ylabel('X Position [m]')
    ax2.legend(); ax2.grid(True)

    plt.tight_layout()
```

```
    plt.show()

# ==============================================================================
# 2. FINÁLNÍ SOUHRN
# ==============================================================================
print("\n" + "="*80)
print(f"PRŮMĚRNÉ VÝSLEDKY ({len(test_data_raw)} trajektorií)")
print("="*80)
print(f"{'Metoda':<10} | {'MSE':<12} | {'RMSE':<12} | {'ANEES':<12}")
print("-" * 80)
print(f"{'BKN':<10} | {np.nanmean(results['BKN_MSE']):<12.2f} | {np.sqrt(np.
 ↪nanmean(results['BKN_MSE'])):<12.2f} | {np.nanmean(results['BKN_ANEES']):<12.
 ↪2f}")
print(f"{'EKF':<10} | {np.nanmean(results['EKF_MSE']):<12.2f} | {np.sqrt(np.
 ↪nanmean(results['EKF_MSE'])):<12.2f} | {np.nanmean(results['EKF_ANEES']):<12.
 ↪2f}")
print("="*80)
```

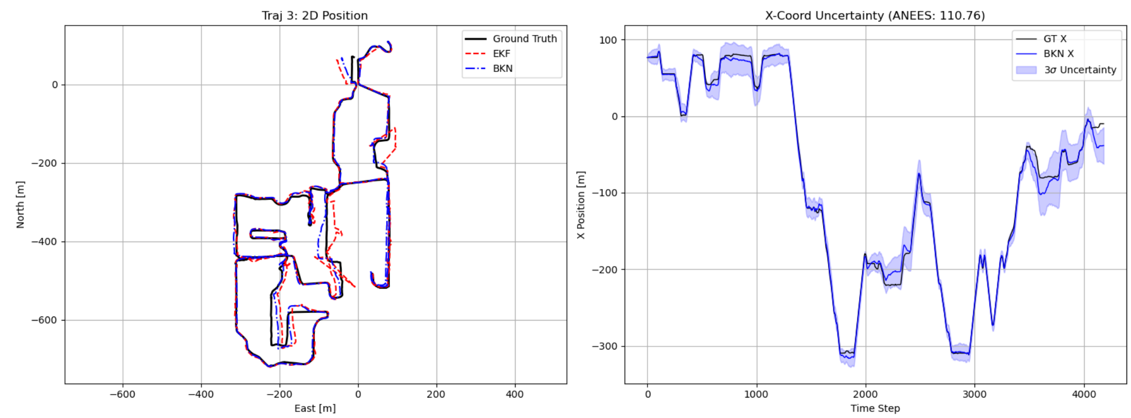
```
INFO: Nastaveno sys_model.dt = 1.0 s
INFO: Používám Bayesian KalmanNet s J=50 vzorky.

Spouštím detailní evaluaci na 3 trajektoriích…

--------------------------------------------------------------------------------
---------------
TRAJEKTORIE   | BKN MSE     | BKN ANEES   | EKF MSE     | EKF ANEES   | GPS MSE
--------------------------------------------------------------------------------
---------------
1             | 248.33      | 32.18       | 443.11      | 1027.02     | 164.46
```




```
2             | 46.36       | 45.53       | 221.68      | 445.10      | 51.13
```




```
3                | 40.44         | 110.76        | 128.44        | 311.25        | 39.94
```




```
================================================================================
PRŮMĚRNÉ VÝSLEDKY (3 trajektorií)
================================================================================
Metoda      | MSE            | RMSE           | ANEES
--------------------------------------------------------------------------------
BKN         | 111.71         | 10.57          | 62.82
EKF         | 264.41         | 16.26          | 594.46
================================================================================
```

[ ]:
```python
# import pandas as pd
# import numpy as np
# import torch

# # 1. Načtení první trajektorie
# traj_data = test_data_raw[0]
```

```
# #␣
↪=============================================================================
# # NOVÁ ČÁST: Průzkum struktury dat
# #␣
↪=============================================================================
# print(f"--- STRUKTURA TRAJEKTORIE (Dostupné klíče) ---")
# print(f"{'Název klíče':<20} | {'Typ':<15} | {'Shape / Hodnota'}")
# print("-" * 60)

# for key, value in traj_data.items():
#     if torch.is_tensor(value):
#         info = f"Tensor {list(value.shape)}"
#     elif isinstance(value, np.ndarray):
#         info = f"NumPy {value.shape}"
#     elif isinstance(value, list):
#         info = f"List (len={len(value)})"
#     else:
#         info = str(value) # Pro stringy, čísla atd.

#     print(f"{key:<20} | {type(value).__name__:<15} | {info}")
# print("=" * 60)
# #␣
↪=============================================================================

# N_ROWS = 500

# # 2. Extrakce dat na CPU
# # Převedeme tensory na numpy pole pro Pandas
# gt = traj_data['ground_truth'][:N_ROWS].cpu().numpy()       # [x, y, theta]
# gps = traj_data['filtered_gps'][:N_ROWS].cpu().numpy()      # [x, y]
# odo = traj_data['filtered_wheel'][:N_ROWS].cpu().numpy()    # [v_left,␣
↪v_right]
# imu = traj_data['imu'][:N_ROWS].cpu().numpy()               # [acc/gyro...]

# # 3. Zpracování ČASU (Klíčové pro EKF!)
# # NCLT 'data_date' bývá v mikrosekundách (utime)
# # Zkontrolujeme, zda existuje klíč 'data_date', jinak fallback
# if 'data_date' in traj_data:
#     raw_time = traj_data['data_date'][:N_ROWS]
# else:
#     # Pokud data_date chybí, zkusíme 'time' nebo vyrobíme dummy čas
#     raw_time = np.arange(N_ROWS)

# # Převedeme na float pole
# if isinstance(raw_time, list):
#     t_vals = np.array([float(t) for t in raw_time])
```

```
# elif isinstance(raw_time, torch.Tensor):
#     t_vals = raw_time.float().cpu().numpy()
# else:
#     t_vals = np.arange(N_ROWS) * 1.0

# # Výpočet DT (rozdíl času oproti předchozímu řádku)
# dt_vals = np.zeros_like(t_vals)
# dt_vals[1:] = t_vals[1:] - t_vals[:-1]

# # Detekce jednotek: Pokud je průměrné dt > 1000, jsou to zřejmě mikrosekundy␣
 ↪-> převedeme na sekundy
# time_unit = "sec"
# if np.mean(dt_vals[1:]) > 1000:
#     dt_vals = dt_vals / 1e6
#     time_unit = "µs -> sec"

# # 4. Sestavení Tabulky
# # Pozor: Zde se snažíme přistoupit ke sloupcům, které existují.
# # Pokud IMU má méně sloupců, pandas vyhodí chybu. Přidám kontrolu shape.
# df_data = {
#     'Time_Step': range(N_ROWS),
#     f'dt [{time_unit}]': dt_vals,
#     'GT_X': gt[:, 0],
#     'GT_Y': gt[:, 1],
#     'GPS_X': gps[:, 0],
#     'GPS_Y': gps[:, 1],
#     'Wheel_L': odo[:, 0],
#     'Wheel_R': odo[:, 1],
# }

# # Dynamické přidání IMU sloupců podle toho, kolik jich tam je
# for i in range(imu.shape[1]):
#     df_data[f'IMU_{i}'] = imu[:, i]

# df = pd.DataFrame(df_data)

# # 5. Výpis statistik pro kontrolu
# print(f"\n--- ANALÝZA DAT (Prvních {N_ROWS} řádků) ---")
# print(f"Průměrné dt: {np.mean(dt_vals[1:]):.5f} s")
# print(f"Max dt:      {np.max(dt_vals[1:]):.5f} s")
# print(f"Počet NaN v GPS_X: {df['GPS_X'].isna().sum()} / {N_ROWS}")
# print("-" * 50)

# # Zobrazí barevnou tabulku (pokud jsi v Jupyteru/Colab)
# def highlight_nan(row):
#     if pd.isna(row['GPS_X']):
#         return ['background-color: #ffcccc'] * len(row)
```

```
#     else:
#         return [''] * len(row)

# # Vykreslení
# try:
#     display(df.style.apply(highlight_nan, axis=1))
# except NameError:
#     print(df.head(20))
```