main_runner_linear_system_trajectory_wise

September 29, 2025

[1]:
```
import sys
import os

notebook_path = os.getcwd()
parent_dir = os.path.dirname(notebook_path)
project_root = os.path.dirname(parent_dir)
if project_root not in sys.path:
    sys.path.insert(0, project_root)
```

[2]:
```
import torch
import torch.nn as nn
from torch.utils.data import DataLoader, TensorDataset
import matplotlib.pyplot as plt
from copy import deepcopy
```

[3]:
```
import state_NN_models
import Filters
import utils
import Systems
from utils import losses, trainer, utils
from torch.utils.data import TensorDataset, DataLoader, random_split
from state_NN_models.StateBayesianKalmanNet import StateBayesianKalmanNet
from state_NN_models.StateKalmanNet import StateKalmanNet
from state_NN_models.StateKalmanNetWithKnownR import StateKalmanNetWithKnownR
```

[4]:
```
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print(f"Používané zařízení: {device}")
```

```
Používané zařízení: cuda
```

[5]:
```
state_dim_2d = 2
obs_dim_2d = 2

F_base_2d = torch.tensor([[1.0, 1.0],
                          [0.0, 1.0]])

svd_F = torch.linalg.svd(F_base_2d)
F_true_2d = F_base_2d / svd_F.S[0]
```

```
H_true_2d = torch.eye(obs_dim_2d)

Q_true_2d = torch.eye(state_dim_2d) * 0.5 # Šum procesu
R_true_2d = torch.eye(obs_dim_2d) * 0.1 # Šum měření

# Počáteční podmínky
Ex0_true_2d = torch.tensor([[1.0], [0.0]])
P0_true_2d = torch.eye(state_dim_2d) * 1.5
F_model_2d = F_true_2d
H_model_2d = H_true_2d
Q_model_2d = torch.eye(state_dim_2d) * 0.1
R_model_2d = R_true_2d
Ex0_model_2d = torch.tensor([[0.5], [0.5]])
P0_model_2d = torch.eye(state_dim_2d) * 1.0

print("\nInicializuji 2D Linear_Canonical systém (replikace autorů)...")
sys_true = Systems.DynamicSystem(
    state_dim=state_dim_2d, obs_dim=obs_dim_2d,
    Ex0=Ex0_true_2d, P0=P0_true_2d,
    Q=Q_true_2d, R=R_true_2d,
    F=F_true_2d, H=H_true_2d,
    device=device
)
sys_model = Systems.DynamicSystem(
    state_dim=state_dim_2d, obs_dim=obs_dim_2d,
    Ex0=Ex0_model_2d, P0=P0_model_2d,
    Q=Q_model_2d, R=R_model_2d,
    F=F_model_2d, H=H_model_2d,
    device=device
)
print("... 2D systém inicializován.")
```

```
Inicializuji 2D Linear_Canonical systém (replikace autorů)…
… 2D systém inicializován.
```

[6]:
```
TRAIN_SEQ_LEN = 10      # Krátké sekvence pro stabilní trénink (TBPTT)
VALID_SEQ_LEN = 20      # Stejná délka pro konzistentní validaci
TEST_SEQ_LEN = 100      # Dlouhé sekvence pro testování generalizace

NUM_TRAIN_TRAJ = 500   # Hodně trénovacích příkladů
NUM_VALID_TRAJ = 200    # Dostatek pro spolehlivou validaci
NUM_TEST_TRAJ = 100     # Pro robustní vyhodnocení

BATCH_SIZE = 8         # Dobrý kompromis
```

```
x_train, y_train = utils.generate_data(sys_true,␣
 ↪num_trajectories=NUM_TRAIN_TRAJ, seq_len=TRAIN_SEQ_LEN)
x_val, y_val = utils.generate_data(sys_true, num_trajectories=NUM_VALID_TRAJ,␣
 ↪seq_len=VALID_SEQ_LEN)
x_test, y_test = utils.generate_data(sys_true, num_trajectories=1,␣
 ↪seq_len=TEST_SEQ_LEN)

train_dataset = TensorDataset(x_train, y_train)
val_dataset = TensorDataset(x_val, y_val)

train_loader = DataLoader(train_dataset, batch_size=BATCH_SIZE, shuffle=True)
val_loader = DataLoader(val_dataset, batch_size=BATCH_SIZE, shuffle=False)
```

[7]:
```
import torch
import torch.nn as nn
from torch.utils.data import TensorDataset, DataLoader
import numpy as np
import os
import random
import csv
from datetime import datetime
import pandas as pd
from copy import deepcopy

model_config = {
    "hidden_size_multiplier": 10,
    "output_layer_multiplier": 4,
    "num_gru_layers": 1,
    "init_min_dropout": 0.5,
    "init_max_dropout": 0.8
}

train_config = {
    "total_train_iter": 1200,
    "learning_rate": 1e-4,
    "clip_grad": 10.0,
    "J_samples": 20,
    "validation_period": 20,
    "logging_period": 20,
    "warmup_iterations":100 # Trénuj prvních 400 iterací jen na MSE
}

#␣
 ↪============================================================================
# KROK 3: SPUŠTĚNÍ JEDNOHO TRÉNINKOVÉHO BĚHU
#␣
 ↪============================================================================
```

```
print("="*80)
print("Spouštím jeden plnohodnotný tréninkový běh...")
print(f"Parametry modelu: {model_config}")
print(f"Parametry tréninku: {train_config}")
print("="*80)

# Nastavení seedu pro reprodukovatelnost tohoto běhu
torch.manual_seed(42)
np.random.seed(42)
random.seed(42)

# Vytvoření modelu
state_bkn_knet = StateBayesianKalmanNet(
    sys_model,
    device=device,
    **model_config
).to(device)

# Spuštění tréninku
# Používáme `run_training_session`, která vrací slovník s výsledky
results = trainer.
 ↪training_session_trajectory_with_gaussian_nll_training_fcn(model=state_bkn_knet,
    train_loader=train_loader,
    val_loader=val_loader,
    device=device,
    **train_config
)

# `run_training_session` automaticky načte nejlepší model zpět,
# takže `state_bkn_knet` nyní obsahuje váhy nejlepšího modelu.
trained_model = results['final_model']

print("\n" + "="*80)
print("TRÉNINK DOKONČEN - FINÁLNÍ VÝSLEDKY Z NEJLEPŠÍHO MODELU")
print("="*80)
print(f"Nejlepší model byl nalezen v iteraci: {results['best_iter']}")
# --- Změněné klíče, aby odpovídaly return statementu ---
print(f"Nejlepší dosažený validační ANEES: {results['best_val_anees']:.4f}")
print("--- Metriky odpovídající tomuto nejlepšímu modelu ---")
print(f"  MSE na validační sadě:       {results['best_val_mse']:.4f}")
print(f"  NLL na validační sadě:       {results['best_val_nll']:.4f}")
print("="*80)

# Nyní můžeš s `trained_model` pokračovat, například ho vyhodnotit na testovací␣
 ↪sadě.
```

```
================================================================================
Spouštím jeden plnohodnotný tréninkový běh…
Parametry modelu: {'hidden_size_multiplier': 10, 'output_layer_multiplier': 4,
'num_gru_layers': 1, 'init_min_dropout': 0.5, 'init_max_dropout': 0.8}
Parametry tréninku: {'total_train_iter': 1200, 'learning_rate': 0.0001,
'clip_grad': 10.0, 'J_samples': 20, 'validation_period': 20, 'logging_period':
20, 'warmup_iterations': 100}
================================================================================
--- Iteration [20/1200] ---
    - Total Loss: 0.3362
    - NLL: 0.0000
    - Reg: 0.0026
    - p1=0.602, p2=0.654

--- Validace v iteraci 20 ---
  Průměrný MSE: 0.3805, Průměrný ANEES: 27.1044
  >>> Nové nejlepší VALIDAČNÍ ANEES! Ukládám model. <<<
------------------------------------------------
--- Iteration [40/1200] ---
    - Total Loss: 0.2803
    - NLL: 0.0000
    - Reg: 0.0026
    - p1=0.602, p2=0.654

--- Validace v iteraci 40 ---
  Průměrný MSE: 0.2710, Průměrný ANEES: 23.7948
  >>> Nové nejlepší VALIDAČNÍ ANEES! Ukládám model. <<<
------------------------------------------------
--- Iteration [60/1200] ---
    - Total Loss: 0.1560
    - NLL: 0.0000
    - Reg: 0.0026
    - p1=0.602, p2=0.654

--- Validace v iteraci 60 ---
  Průměrný MSE: 0.2044, Průměrný ANEES: 18.6005
  >>> Nové nejlepší VALIDAČNÍ ANEES! Ukládám model. <<<
------------------------------------------------
--- Iteration [80/1200] ---
    - Total Loss: 0.1446
    - NLL: 0.0000
    - Reg: 0.0026
    - p1=0.602, p2=0.653

--- Validace v iteraci 80 ---
  Průměrný MSE: 0.1608, Průměrný ANEES: 16.7818
  >>> Nové nejlepší VALIDAČNÍ ANEES! Ukládám model. <<<
------------------------------------------------
```

```
--- Iteration [100/1200] ---
    - Total Loss: 0.1126
    - NLL: 0.0000
    - Reg: 0.0026
    - p1=0.601, p2=0.653

--- Validace v iteraci 100 ---
  Průměrný MSE: 0.1342, Průměrný ANEES: 14.6854
  >>> Nové nejlepší VALIDAČNÍ ANEES! Ukládám model. <<<
--------------------------------------------------
--- Iteration [120/1200] ---
    - Total Loss: 2.2789
    - NLL: 2.2763
    - Reg: 0.0026
    - p1=0.602, p2=0.654

--- Validace v iteraci 120 ---
  Průměrný MSE: 0.1110, Průměrný ANEES: 15.1610
--------------------------------------------------
--- Iteration [140/1200] ---
    - Total Loss: 0.9203
    - NLL: 0.9177
    - Reg: 0.0026
    - p1=0.602, p2=0.654

--- Validace v iteraci 140 ---
  Průměrný MSE: 0.0986, Průměrný ANEES: 11.7256
  >>> Nové nejlepší VALIDAČNÍ ANEES! Ukládám model. <<<
--------------------------------------------------
--- Iteration [160/1200] ---
    - Total Loss: 0.8684
    - NLL: 0.8657
    - Reg: 0.0026
    - p1=0.603, p2=0.655

--- Validace v iteraci 160 ---
  Průměrný MSE: 0.0967, Průměrný ANEES: 8.4057
  >>> Nové nejlepší VALIDAČNÍ ANEES! Ukládám model. <<<
--------------------------------------------------
--- Iteration [180/1200] ---
    - Total Loss: 1.3624
    - NLL: 1.3598
    - Reg: 0.0027
    - p1=0.603, p2=0.656

--- Validace v iteraci 180 ---
  Průměrný MSE: 0.0953, Průměrný ANEES: 7.4388
  >>> Nové nejlepší VALIDAČNÍ ANEES! Ukládám model. <<<
```

```
------------------------------------------------
--- Iteration [200/1200] ---
    - Total Loss: 1.0595
    - NLL: 1.0569
    - Reg: 0.0027
    - p1=0.603, p2=0.656

--- Validace v iteraci 200 ---
  Průměrný MSE: 0.0953, Průměrný ANEES: 6.4499
  >>> Nové nejlepší VALIDAČNÍ ANEES! Ukládám model. <<<
------------------------------------------------
--- Iteration [220/1200] ---
    - Total Loss: 1.1729
    - NLL: 1.1703
    - Reg: 0.0027
    - p1=0.603, p2=0.656

--- Validace v iteraci 220 ---
  Průměrný MSE: 0.0976, Průměrný ANEES: 7.9357
------------------------------------------------
--- Iteration [240/1200] ---
    - Total Loss: 0.7544
    - NLL: 0.7517
    - Reg: 0.0027
    - p1=0.603, p2=0.656

--- Validace v iteraci 240 ---
  Průměrný MSE: 0.0961, Průměrný ANEES: 6.5721
------------------------------------------------
--- Iteration [260/1200] ---
    - Total Loss: 0.8057
    - NLL: 0.8030
    - Reg: 0.0027
    - p1=0.603, p2=0.657

--- Validace v iteraci 260 ---
  Průměrný MSE: 0.0989, Průměrný ANEES: 5.8726
  >>> Nové nejlepší VALIDAČNÍ ANEES! Ukládám model. <<<
------------------------------------------------
--- Iteration [280/1200] ---
    - Total Loss: 1.4575
    - NLL: 1.4548
    - Reg: 0.0027
    - p1=0.604, p2=0.657

--- Validace v iteraci 280 ---
  Průměrný MSE: 0.0981, Průměrný ANEES: 5.6881
  >>> Nové nejlepší VALIDAČNÍ ANEES! Ukládám model. <<<
```

```
------------------------------------------------
--- Iteration [300/1200] ---
    - Total Loss: 1.0165
    - NLL: 1.0138
    - Reg: 0.0027
    - p1=0.604, p2=0.657

--- Validace v iteraci 300 ---
  Průměrný MSE: 0.0972, Průměrný ANEES: 5.4375
  >>> Nové nejlepší VALIDAČNÍ ANEES! Ukládám model. <<<
------------------------------------------------
--- Iteration [320/1200] ---
    - Total Loss: 1.3337
    - NLL: 1.3311
    - Reg: 0.0027
    - p1=0.604, p2=0.657

--- Validace v iteraci 320 ---
  Průměrný MSE: 0.0962, Průměrný ANEES: 5.1504
  >>> Nové nejlepší VALIDAČNÍ ANEES! Ukládám model. <<<
------------------------------------------------
--- Iteration [340/1200] ---
    - Total Loss: 1.7900
    - NLL: 1.7873
    - Reg: 0.0027
    - p1=0.604, p2=0.657

--- Validace v iteraci 340 ---
  Průměrný MSE: 0.0969, Průměrný ANEES: 5.6892
------------------------------------------------
--- Iteration [360/1200] ---
    - Total Loss: 1.2782
    - NLL: 1.2756
    - Reg: 0.0027
    - p1=0.604, p2=0.657

--- Validace v iteraci 360 ---
  Průměrný MSE: 0.0983, Průměrný ANEES: 4.9351
  >>> Nové nejlepší VALIDAČNÍ ANEES! Ukládám model. <<<
------------------------------------------------
--- Iteration [380/1200] ---
    - Total Loss: 0.6425
    - NLL: 0.6398
    - Reg: 0.0027
    - p1=0.604, p2=0.657

--- Validace v iteraci 380 ---
  Průměrný MSE: 0.0976, Průměrný ANEES: 5.1032
```

```
------------------------------------------------
--- Iteration [400/1200] ---
    - Total Loss: 1.2036
    - NLL: 1.2009
    - Reg: 0.0027
    - p1=0.604, p2=0.658

--- Validace v iteraci 400 ---
  Průměrný MSE: 0.0983, Průměrný ANEES: 7.2309
------------------------------------------------
--- Iteration [420/1200] ---
    - Total Loss: 0.8077
    - NLL: 0.8050
    - Reg: 0.0027
    - p1=0.604, p2=0.658

--- Validace v iteraci 420 ---
  Průměrný MSE: 0.0988, Průměrný ANEES: 4.5952
  >>> Nové nejlepší VALIDAČNÍ ANEES! Ukládám model. <<<
------------------------------------------------
--- Iteration [440/1200] ---
    - Total Loss: 1.0864
    - NLL: 1.0837
    - Reg: 0.0027
    - p1=0.604, p2=0.658

--- Validace v iteraci 440 ---
  Průměrný MSE: 0.0983, Průměrný ANEES: 4.9415
------------------------------------------------
--- Iteration [460/1200] ---
    - Total Loss: 1.1810
    - NLL: 1.1784
    - Reg: 0.0027
    - p1=0.604, p2=0.658

--- Validace v iteraci 460 ---
  Průměrný MSE: 0.0981, Průměrný ANEES: 4.7117
------------------------------------------------
--- Iteration [480/1200] ---
    - Total Loss: 1.0330
    - NLL: 1.0304
    - Reg: 0.0027
    - p1=0.604, p2=0.658

--- Validace v iteraci 480 ---
  Průměrný MSE: 0.0968, Průměrný ANEES: 5.5896
------------------------------------------------
--- Iteration [500/1200] ---
```

```
    - Total Loss: 1.3101
    - NLL: 1.3075
    - Reg: 0.0027
    - p1=0.604, p2=0.658

--- Validace v iteraci 500 ---
  Průměrný MSE: 0.0978, Průměrný ANEES: 4.5177
  >>> Nové nejlepší VALIDAČNÍ ANEES! Ukládám model. <<<
--------------------------------------------------
--- Iteration [520/1200] ---
    - Total Loss: 0.8979
    - NLL: 0.8952
    - Reg: 0.0027
    - p1=0.604, p2=0.658

--- Validace v iteraci 520 ---
  Průměrný MSE: 0.1009, Průměrný ANEES: 4.5452
--------------------------------------------------
--- Iteration [540/1200] ---
    - Total Loss: 0.6790
    - NLL: 0.6764
    - Reg: 0.0027
    - p1=0.604, p2=0.658

--- Validace v iteraci 540 ---
  Průměrný MSE: 0.0999, Průměrný ANEES: 4.2554
  >>> Nové nejlepší VALIDAČNÍ ANEES! Ukládám model. <<<
--------------------------------------------------
--- Iteration [560/1200] ---
    - Total Loss: 1.0580
    - NLL: 1.0554
    - Reg: 0.0027
    - p1=0.605, p2=0.658

--- Validace v iteraci 560 ---
  Průměrný MSE: 0.0985, Průměrný ANEES: 4.1195
  >>> Nové nejlepší VALIDAČNÍ ANEES! Ukládám model. <<<
--------------------------------------------------
--- Iteration [580/1200] ---
    - Total Loss: 2.1050
    - NLL: 2.1024
    - Reg: 0.0027
    - p1=0.604, p2=0.658

--- Validace v iteraci 580 ---
  Průměrný MSE: 0.0996, Průměrný ANEES: 5.0714
--------------------------------------------------
--- Iteration [600/1200] ---
```

```
    - Total Loss: 0.5786
    - NLL: 0.5759
    - Reg: 0.0027
    - p1=0.604, p2=0.658

--- Validace v iteraci 600 ---
  Průměrný MSE: 0.0978, Průměrný ANEES: 4.5530
--------------------------------------------------
--- Iteration [620/1200] ---
    - Total Loss: 0.9613
    - NLL: 0.9586
    - Reg: 0.0027
    - p1=0.605, p2=0.658

--- Validace v iteraci 620 ---
  Průměrný MSE: 0.0992, Průměrný ANEES: 4.8813
--------------------------------------------------
--- Iteration [640/1200] ---
    - Total Loss: 1.5945
    - NLL: 1.5919
    - Reg: 0.0027
    - p1=0.605, p2=0.658

--- Validace v iteraci 640 ---
  Průměrný MSE: 0.0982, Průměrný ANEES: 5.3026
--------------------------------------------------
--- Iteration [660/1200] ---
    - Total Loss: 1.0652
    - NLL: 1.0625
    - Reg: 0.0027
    - p1=0.604, p2=0.658

--- Validace v iteraci 660 ---
  Průměrný MSE: 0.0990, Průměrný ANEES: 4.5190
--------------------------------------------------
--- Iteration [680/1200] ---
    - Total Loss: 0.6405
    - NLL: 0.6378
    - Reg: 0.0027
    - p1=0.604, p2=0.658

--- Validace v iteraci 680 ---
  Průměrný MSE: 0.0993, Průměrný ANEES: 5.2844
--------------------------------------------------
--- Iteration [700/1200] ---
    - Total Loss: 1.0157
    - NLL: 1.0131
    - Reg: 0.0027
```

```
    - p1=0.604, p2=0.658

--- Validace v iteraci 700 ---
  Průměrný MSE: 0.1009, Průměrný ANEES: 5.0288
--------------------------------------------------
--- Iteration [720/1200] ---
    - Total Loss: 1.1495
    - NLL: 1.1468
    - Reg: 0.0027
    - p1=0.604, p2=0.658

--- Validace v iteraci 720 ---
  Průměrný MSE: 0.0999, Průměrný ANEES: 4.4257
--------------------------------------------------
--- Iteration [740/1200] ---
    - Total Loss: 0.6194
    - NLL: 0.6167
    - Reg: 0.0027
    - p1=0.604, p2=0.658

--- Validace v iteraci 740 ---
  Průměrný MSE: 0.1009, Průměrný ANEES: 5.0289
--------------------------------------------------
--- Iteration [760/1200] ---
    - Total Loss: 0.8556
    - NLL: 0.8529
    - Reg: 0.0027
    - p1=0.604, p2=0.658

--- Validace v iteraci 760 ---
  Průměrný MSE: 0.0990, Průměrný ANEES: 4.4624
--------------------------------------------------
--- Iteration [780/1200] ---
    - Total Loss: 1.1400
    - NLL: 1.1373
    - Reg: 0.0027
    - p1=0.604, p2=0.658

--- Validace v iteraci 780 ---
  Průměrný MSE: 0.1013, Průměrný ANEES: 4.7345
--------------------------------------------------
--- Iteration [800/1200] ---
    - Total Loss: 1.4221
    - NLL: 1.4195
    - Reg: 0.0027
    - p1=0.604, p2=0.658

--- Validace v iteraci 800 ---
```

```
  Průměrný MSE: 0.1004, Průměrný ANEES: 4.7233
--------------------------------------------------
--- Iteration [820/1200] ---
    - Total Loss: 1.2151
    - NLL: 1.2125
    - Reg: 0.0027
    - p1=0.604, p2=0.658

--- Validace v iteraci 820 ---
  Průměrný MSE: 0.0990, Průměrný ANEES: 5.6339
--------------------------------------------------
--- Iteration [840/1200] ---
    - Total Loss: 0.7458
    - NLL: 0.7432
    - Reg: 0.0027
    - p1=0.604, p2=0.658

--- Validace v iteraci 840 ---
  Průměrný MSE: 0.1012, Průměrný ANEES: 6.0197
--------------------------------------------------
--- Iteration [860/1200] ---
    - Total Loss: 0.8455
    - NLL: 0.8428
    - Reg: 0.0027
    - p1=0.604, p2=0.658

--- Validace v iteraci 860 ---
  Průměrný MSE: 0.1012, Průměrný ANEES: 4.2020
--------------------------------------------------
--- Iteration [880/1200] ---
    - Total Loss: 1.3893
    - NLL: 1.3866
    - Reg: 0.0027
    - p1=0.604, p2=0.658

--- Validace v iteraci 880 ---
  Průměrný MSE: 0.0998, Průměrný ANEES: 5.3442
--------------------------------------------------
--- Iteration [900/1200] ---
    - Total Loss: 1.0787
    - NLL: 1.0761
    - Reg: 0.0027
    - p1=0.604, p2=0.658

--- Validace v iteraci 900 ---
  Průměrný MSE: 0.1015, Průměrný ANEES: 4.1744
--------------------------------------------------
--- Iteration [920/1200] ---
```

```
    - Total Loss: 0.8513
    - NLL: 0.8486
    - Reg: 0.0027
    - p1=0.604, p2=0.658

--- Validace v iteraci 920 ---
  Průměrný MSE: 0.0990, Průměrný ANEES: 5.7691
------------------------------------------------
--- Iteration [940/1200] ---
    - Total Loss: 0.7088
    - NLL: 0.7061
    - Reg: 0.0027
    - p1=0.604, p2=0.658

--- Validace v iteraci 940 ---
  Průměrný MSE: 0.0981, Průměrný ANEES: 4.3610
------------------------------------------------
--- Iteration [960/1200] ---
    - Total Loss: 0.6808
    - NLL: 0.6781
    - Reg: 0.0027
    - p1=0.604, p2=0.658

--- Validace v iteraci 960 ---
  Průměrný MSE: 0.0997, Průměrný ANEES: 4.2159
------------------------------------------------
--- Iteration [980/1200] ---
    - Total Loss: 1.0023
    - NLL: 0.9996
    - Reg: 0.0027
    - p1=0.604, p2=0.658

--- Validace v iteraci 980 ---
  Průměrný MSE: 0.1002, Průměrný ANEES: 4.2842
------------------------------------------------
--- Iteration [1000/1200] ---
    - Total Loss: 0.6944
    - NLL: 0.6918
    - Reg: 0.0027
    - p1=0.604, p2=0.658

--- Validace v iteraci 1000 ---
  Průměrný MSE: 0.0997, Průměrný ANEES: 5.9774
------------------------------------------------
--- Iteration [1020/1200] ---
    - Total Loss: 0.9048
    - NLL: 0.9021
    - Reg: 0.0027
```

```
    - p1=0.604, p2=0.658

--- Validace v iteraci 1020 ---
  Průměrný MSE: 0.1003, Průměrný ANEES: 4.4544
--------------------------------------------------
--- Iteration [1040/1200] ---
    - Total Loss: 1.1099
    - NLL: 1.1073
    - Reg: 0.0027
    - p1=0.604, p2=0.658

--- Validace v iteraci 1040 ---
  Průměrný MSE: 0.0997, Průměrný ANEES: 4.0311
  >>> Nové nejlepší VALIDAČNÍ ANEES! Ukládám model. <<<
--------------------------------------------------
--- Iteration [1060/1200] ---
    - Total Loss: 0.9737
    - NLL: 0.9710
    - Reg: 0.0027
    - p1=0.604, p2=0.658

--- Validace v iteraci 1060 ---
  Průměrný MSE: 0.1000, Průměrný ANEES: 4.7272
--------------------------------------------------
--- Iteration [1080/1200] ---
    - Total Loss: 1.7426
    - NLL: 1.7400
    - Reg: 0.0027
    - p1=0.604, p2=0.658

--- Validace v iteraci 1080 ---
  Průměrný MSE: 0.1004, Průměrný ANEES: 4.9461
--------------------------------------------------
--- Iteration [1100/1200] ---
    - Total Loss: 0.4802
    - NLL: 0.4775
    - Reg: 0.0027
    - p1=0.604, p2=0.658

--- Validace v iteraci 1100 ---
  Průměrný MSE: 0.0991, Průměrný ANEES: 5.2526
--------------------------------------------------
--- Iteration [1120/1200] ---
    - Total Loss: 1.0493
    - NLL: 1.0466
    - Reg: 0.0027
    - p1=0.604, p2=0.658
```

```
--- Validace v iteraci 1120 ---
  Průměrný MSE: 0.0983, Průměrný ANEES: 4.3582
------------------------------------------------
--- Iteration [1140/1200] ---
    - Total Loss: 0.6733
    - NLL: 0.6707
    - Reg: 0.0027
    - p1=0.604, p2=0.658

--- Validace v iteraci 1140 ---
  Průměrný MSE: 0.0996, Průměrný ANEES: 5.2705
------------------------------------------------
--- Iteration [1160/1200] ---
    - Total Loss: 0.8339
    - NLL: 0.8312
    - Reg: 0.0027
    - p1=0.604, p2=0.658

--- Validace v iteraci 1160 ---
  Průměrný MSE: 0.0995, Průměrný ANEES: 4.6762
------------------------------------------------
--- Iteration [1180/1200] ---
    - Total Loss: 0.7277
    - NLL: 0.7251
    - Reg: 0.0027
    - p1=0.604, p2=0.658

--- Validace v iteraci 1180 ---
  Průměrný MSE: 0.0983, Průměrný ANEES: 5.2171
------------------------------------------------
--- Iteration [1200/1200] ---
    - Total Loss: 1.0441
    - NLL: 1.0415
    - Reg: 0.0027
    - p1=0.604, p2=0.658

--- Validace v iteraci 1200 ---
  Průměrný MSE: 0.0989, Průměrný ANEES: 4.7241
------------------------------------------------

Trénování dokončeno.
Načítám nejlepší model z iterace 1040 s ANEES 4.0311

================================================================================
TRÉNINK DOKONČEN - FINÁLNÍ VÝSLEDKY Z NEJLEPŠÍHO MODELU
================================================================================
Nejlepší model byl nalezen v iteraci: 1040
Nejlepší dosažený validační ANEES: 4.0311
```

```
--- Metriky odpovídající tomuto nejlepšímu modelu ---
  MSE na validační sadě:       0.0997
  NLL na validační sadě:       0.0000
================================================================================
```

[8]:
```python
import torch
import torch.nn as nn
from torch.utils.data import TensorDataset, DataLoader
import numpy as np
import os
import random
import csv
from datetime import datetime
import pandas as pd
from copy import deepcopy
# Nastavení seedu pro reprodukovatelnost tohoto běhu
torch.manual_seed(42)
np.random.seed(42)
random.seed(42)
state_knet = StateKalmanNet(sys_model, device=device,␣
 ↪hidden_size_multiplier=12).to(device)
trainer.train_state_KalmanNet(
    model=state_knet,
    train_loader=train_loader,
    val_loader=val_loader,
    device=device,
    epochs=100,
    lr=1e-4,
    early_stopping_patience=30
)
```

```
INFO: Detekováno, že model vrací kovarianci: False

/home/luky/.local/lib/python3.10/site-packages/torch/optim/lr_scheduler.py:28:
UserWarning: The verbose parameter is deprecated. Please use get_last_lr() to
access the learning rate.
  warnings.warn("The verbose parameter is deprecated. Please use get_last_lr() "

Epoch [5/100], Train Loss: 0.105876, Val Loss: 0.098402
Epoch [10/100], Train Loss: 0.092966, Val Loss: 0.090529
Epoch [15/100], Train Loss: 0.087301, Val Loss: 0.087353
Epoch [20/100], Train Loss: 0.084825, Val Loss: 0.086125
Epoch [25/100], Train Loss: 0.084149, Val Loss: 0.085853
Epoch [30/100], Train Loss: 0.083553, Val Loss: 0.085639
Epoch [35/100], Train Loss: 0.083446, Val Loss: 0.085613
Epoch [40/100], Train Loss: 0.083388, Val Loss: 0.085493
Epoch [45/100], Train Loss: 0.083063, Val Loss: 0.085449
Epoch [50/100], Train Loss: 0.082814, Val Loss: 0.085608
Epoch [55/100], Train Loss: 0.082489, Val Loss: 0.085488
```

```
Epoch [60/100], Train Loss: 0.082818, Val Loss: 0.085525
Epoch [65/100], Train Loss: 0.082572, Val Loss: 0.085518
Epoch [70/100], Train Loss: 0.082625, Val Loss: 0.085515
Epoch [75/100], Train Loss: 0.082494, Val Loss: 0.085519

Early stopping spuštěno po 75 epochách.
Trénování dokončeno.
Načítám nejlepší model s validační chybou: 0.085449
```

[8]:
```
StateKalmanNet(
  (dnn): DNN_KalmanNet(
    (input_layer): Linear(in_features=4, out_features=96, bias=True)
    (gru): GRU(96, 96)
    (output_layer): Linear(in_features=96, out_features=4, bias=True)
  )
)
```

[9]:
```
import torch
import torch.nn as nn
from torch.utils.data import TensorDataset, DataLoader
import numpy as np
import os
import random
import csv
from datetime import datetime
import pandas as pd
from copy import deepcopy
# Nastavení seedu pro reprodukovatelnost tohoto běhu
torch.manual_seed(42)
np.random.seed(42)
random.seed(42)
state_knetR = StateKalmanNetWithKnownR(sys_model, device=device,␣
 ↪hidden_size_multiplier=12).to(device)
trainer.train_state_KalmanNet(
    model=state_knetR,
    train_loader=train_loader,
    val_loader=val_loader,
    device=device,
    epochs=100,
    lr=1e-4,
    early_stopping_patience=30
)
```

```
INFO: Detekováno, že model vrací kovarianci: True
Epoch [5/100], Train Loss: 0.105876, Val Loss: 0.098402, Avg Cov Trace: 0.156468
Epoch [10/100], Train Loss: 0.092966, Val Loss: 0.090529, Avg Cov Trace:
0.160472
Epoch [15/100], Train Loss: 0.087301, Val Loss: 0.087353, Avg Cov Trace:
```

```
0.162282
Epoch [20/100], Train Loss: 0.084825, Val Loss: 0.086125, Avg Cov Trace:
0.163968
Epoch [25/100], Train Loss: 0.084149, Val Loss: 0.085853, Avg Cov Trace:
0.165014
Epoch [30/100], Train Loss: 0.083553, Val Loss: 0.085639, Avg Cov Trace:
0.165752
Epoch [35/100], Train Loss: 0.083446, Val Loss: 0.085613, Avg Cov Trace:
0.165686
Epoch [40/100], Train Loss: 0.083388, Val Loss: 0.085493, Avg Cov Trace:
0.165124
Epoch [45/100], Train Loss: 0.083063, Val Loss: 0.085449, Avg Cov Trace:
0.165539
Epoch [50/100], Train Loss: 0.082814, Val Loss: 0.085608, Avg Cov Trace:
0.166102
Epoch [55/100], Train Loss: 0.082489, Val Loss: 0.085488, Avg Cov Trace:
0.165276
Epoch [60/100], Train Loss: 0.082818, Val Loss: 0.085525, Avg Cov Trace:
0.165254
Epoch [65/100], Train Loss: 0.082572, Val Loss: 0.085518, Avg Cov Trace:
0.165364
Epoch [70/100], Train Loss: 0.082625, Val Loss: 0.085515, Avg Cov Trace:
0.165383
Epoch [75/100], Train Loss: 0.082494, Val Loss: 0.085519, Avg Cov Trace:
0.165441

Early stopping spuštěno po 75 epochách.
Trénování dokončeno.
Načítám nejlepší model s validační chybou: 0.085449
```

[9]:
```
StateKalmanNetWithKnownR(
  (dnn): DNN_KalmanNet(
    (input_layer): Linear(in_features=4, out_features=96, bias=True)
    (gru): GRU(96, 96)
    (output_layer): Linear(in_features=96, out_features=4, bias=True)
  )
)
```

[10]:
```
import torch
import torch.nn.functional as F
import numpy as np
from torch.utils.data import TensorDataset, DataLoader

# ================================================================================
# 0. PŘEDPOKLADY - ZDE PŘIŘAĎTE VAŠE NATRÉNOVANÉ MODELY
# ================================================================================
# Ujistěte se, že v proměnných níže máte již natrénované a připravené modely.
```

```
# Názvy proměnných si upravte podle vašeho kódu, pokud se liší.
try:
    trained_model_bkn = trained_model
    trained_model_classic = state_knet
    trained_model_knetR = state_knetR
    print("INFO: Všechny natrénované modely nalezeny a přiřazeny.")
except NameError:
    print("VAROVÁNÍ: Některé z proměnných `trained_model`, `state_knet`, nebo␣
 ↪`state_knetR` nebyly nalezeny.")
    print("         Ujistěte se, že jste nejprve úspěšně dokončili trénink␣
 ↪všech modelů.")


# ================================================================================
# 1. KONFIGURACE TESTU
# ================================================================================
TEST_SEQ_LEN = 200
NUM_TEST_TRAJ = 20
J_SAMPLES_TEST = 25

# ================================================================================
# 2. PŘÍPRAVA DAT
# ================================================================================
print(f"\nGeneruji {NUM_TEST_TRAJ} testovacích trajektorií o délce␣
 ↪{TEST_SEQ_LEN}...")
x_test, y_test = utils.generate_data(sys_true, num_trajectories=NUM_TEST_TRAJ,␣
 ↪seq_len=TEST_SEQ_LEN)
test_dataset = TensorDataset(x_test, y_test)
test_loader = DataLoader(test_dataset, batch_size=1, shuffle=False)
print("Generování dat dokončeno.")

# ================================================================================
# 3. INICIALIZACE VŠECH FILTRŮ PRO POROVNÁNÍ
# ================================================================================
ekf_mismatched = Filters.ExtendedKalmanFilter(sys_model)
ekf_ideal = Filters.ExtendedKalmanFilter(sys_true)
ukf_mismatched = Filters.UnscentedKalmanFilter(sys_model)
ukf_ideal = Filters.UnscentedKalmanFilter(sys_true)
aekf_mismatched = Filters.AdaptiveExtendedKalmanFilter(sys_model,␣
 ↪Q_init=sys_model.Q, R_init=sys_model.R, alpha=0.999)
print("Všechny model-based filtry (EKF, UKF, AEKF) inicializovány.")

# ================================================================================
# 4. VYHODNOCOVACÍ SMYČKA
# ================================================================================
# Seznamy pro ukládání výsledků z každé trajektorie
all_x_true_cpu = []
```

```
all_x_hat_bkn_cpu, all_P_hat_bkn_cpu = [], []
all_x_hat_classic_knet_cpu = []
all_x_hat_knetR_cpu, all_P_hat_knetR_cpu = [], []
all_x_hat_ekf_mismatched_cpu, all_P_hat_ekf_mismatched_cpu = [], []
all_x_hat_ekf_ideal_cpu, all_P_hat_ekf_ideal_cpu = [], []
all_x_hat_ukf_mismatched_cpu, all_P_hat_ukf_mismatched_cpu = [], []
all_x_hat_ukf_ideal_cpu, all_P_hat_ukf_ideal_cpu = [], []
all_x_hat_aekf_mismatched_cpu, all_P_hat_aekf_mismatched_cpu = [], []

print(f"\nVyhodnocuji modely na {NUM_TEST_TRAJ} testovacích trajektoriích...")

# Důležité: Přepneme všechny NN modely do evaluačního režimu
trained_model_bkn.eval()
trained_model_classic.eval()
trained_model_knetR.eval()

with torch.no_grad():
    for i, (x_true_seq_batch, y_test_seq_batch) in enumerate(test_loader):
        y_test_seq_gpu = y_test_seq_batch.squeeze(0).to(device)
        x_true_seq_gpu = x_true_seq_batch.squeeze(0).to(device)
        initial_state = x_true_seq_gpu[0, :].unsqueeze(0)

        # --- A. Bayesian KalmanNet (Trajectory-wise) ---
        ensemble_trajectories = []
        for j in range(J_SAMPLES_TEST):
            trained_model_bkn.reset(batch_size=1, initial_state=initial_state)
            current_x_hats = []
            for t in range(1, TEST_SEQ_LEN):
                x_filtered_t, _ = trained_model_bkn.step(y_test_seq_gpu[t, :].
 ↪unsqueeze(0))
                current_x_hats.append(x_filtered_t)
            ensemble_trajectories.append(torch.cat(current_x_hats, dim=0))
        ensemble = torch.stack(ensemble_trajectories, dim=0)
        predictions_bkn = ensemble.mean(dim=0)
        diff = ensemble - predictions_bkn.unsqueeze(0)
        covariances_bkn = (diff.unsqueeze(-1) @ diff.unsqueeze(-2)).mean(dim=0)
        full_x_hat_bkn = torch.cat([initial_state, predictions_bkn], dim=0)
        full_P_hat_bkn = torch.cat([sys_model.P0.unsqueeze(0),␣
 ↪covariances_bkn], dim=0)

        # --- B. Klasický StateKalmanNet (pouze MSE) ---
        trained_model_classic.reset(batch_size=1, initial_state=initial_state)
        classic_knet_preds = []
        for t in range(1, TEST_SEQ_LEN):
            x_filtered_t = trained_model_classic.step(y_test_seq_gpu[t, :].
 ↪unsqueeze(0))
            classic_knet_preds.append(x_filtered_t)
```

```
        full_x_hat_classic_knet = torch.cat([initial_state, torch.
↪cat(classic_knet_preds, dim=0)], dim=0)

        # --- C. StateKalmanNetWithKnownR ---
        trained_model_knetR.reset(batch_size=1, initial_state=initial_state)
        knetR_preds_x, knetR_preds_P = [], []
        for t in range(1, TEST_SEQ_LEN):
            x_filtered_t, P_filtered_t = trained_model_knetR.
↪step(y_test_seq_gpu[t, :].unsqueeze(0))
            knetR_preds_x.append(x_filtered_t)
            knetR_preds_P.append(P_filtered_t)
        full_x_hat_knetR = torch.cat([initial_state, torch.cat(knetR_preds_x,␣
↪dim=0)], dim=0)
        processed_P_list = []
        for p_tensor in knetR_preds_P:
            # Zajistíme, aby každý P byl alespoň 2D matice
            while p_tensor.dim() < 2:
                p_tensor = p_tensor.unsqueeze(-1)
            # Odstraníme případnou přebytečnou dávkovou dimenzi z `step` metody
            if p_tensor.dim() > 2 and p_tensor.shape[0] == 1:
                p_tensor = p_tensor.squeeze(0)
            processed_P_list.append(p_tensor)

        # 2. Nyní můžeme bezpečně použít stack
        P_sequence_knetR = torch.stack(processed_P_list, dim=0)

        # 3. Zajistíme, že P0 má také správný počet dimenzí
        P0_for_cat = sys_model.P0.clone()
        while P0_for_cat.dim() < P_sequence_knetR.dim():
            P0_for_cat = P0_for_cat.unsqueeze(0)

        full_P_hat_knetR = torch.cat([P0_for_cat, P_sequence_knetR], dim=0)

        # --- D. EKF (nepřesný a ideální) ---
        ekf_m_res = ekf_mismatched.process_sequence(y_test_seq_gpu,␣
↪Ex0=sys_model.Ex0, P0=sys_model.P0)
        # ŠPATNĚ: full_x_hat_ekf_m = torch.cat([sys_model.Ex0.reshape(1, -1),␣
↪ekf_m_res['x_filtered']], dim=0)
        # SPRÁVNĚ:
        full_x_hat_ekf_m = ekf_m_res['x_filtered'] # Výsledek je již kompletní␣
↪trajektorie
        full_P_hat_ekf_m = ekf_m_res['P_filtered'] # To samé pro kovarianci

        ekf_i_res = ekf_ideal.process_sequence(y_test_seq_gpu, Ex0=sys_true.
↪Ex0, P0=sys_true.P0)
```

```
        # ŠPATNĚ: full_x_hat_ekf_i = torch.cat([sys_true.Ex0.reshape(1, -1),␣
↪ekf_i_res['x_filtered']], dim=0)
        # SPRÁVNĚ:
        full_x_hat_ekf_i = ekf_i_res['x_filtered']
        full_P_hat_ekf_i = ekf_i_res['P_filtered']

        # --- E. UKF (nepřesný a ideální) ---
        ukf_m_res = ukf_mismatched.process_sequence(y_test_seq_gpu,␣
↪Ex0=sys_model.Ex0, P0=sys_model.P0)
        # ŠPATNĚ: full_x_hat_ukf_m = torch.cat([sys_model.Ex0.reshape(1, -1),␣
↪ukf_m_res['x_filtered']], dim=0)
        # SPRÁVNĚ:
        full_x_hat_ukf_m = ukf_m_res['x_filtered']
        full_P_hat_ukf_m = ukf_m_res['P_filtered']

        ukf_i_res = ukf_ideal.process_sequence(y_test_seq_gpu, Ex0=sys_true.
↪Ex0, P0=sys_true.P0)
        # ŠPATNĚ: full_x_hat_ukf_i = torch.cat([sys_true.Ex0.reshape(1, -1),␣
↪ukf_i_res['x_filtered']], dim=0)
        # SPRÁVNĚ:
        full_x_hat_ukf_i = ukf_i_res['x_filtered']
        full_P_hat_ukf_i = ukf_i_res['P_filtered']

        # --- F. Adaptivní EKF (nepřesný) ---
        aekf_m_res = aekf_mismatched.process_sequence(y_test_seq_gpu,␣
↪Ex0=sys_model.Ex0, P0=sys_model.P0)
        # ŠPATNĚ: full_x_hat_aekf_m = torch.cat([sys_model.Ex0.reshape(1, -1),␣
↪aekf_m_res['x_filtered']], dim=0)
        # SPRÁVNĚ:
        full_x_hat_aekf_m = aekf_m_res['x_filtered']
        full_P_hat_aekf_m = aekf_m_res['P_filtered']
        # --- G. Uložení všech výsledků na CPU ---
        all_x_true_cpu.append(x_true_seq_gpu.cpu())
        all_x_hat_bkn_cpu.append(full_x_hat_bkn.cpu()); all_P_hat_bkn_cpu.
↪append(full_P_hat_bkn.cpu())
        all_x_hat_classic_knet_cpu.append(full_x_hat_classic_knet.cpu())
        all_x_hat_knetR_cpu.append(full_x_hat_knetR.cpu()); all_P_hat_knetR_cpu.
↪append(full_P_hat_knetR.cpu())
        all_x_hat_ekf_mismatched_cpu.append(full_x_hat_ekf_m.cpu());␣
↪all_P_hat_ekf_mismatched_cpu.append(full_P_hat_ekf_m.cpu())
        all_x_hat_ekf_ideal_cpu.append(full_x_hat_ekf_i.cpu());␣
↪all_P_hat_ekf_ideal_cpu.append(full_P_hat_ekf_i.cpu())
        all_x_hat_ukf_mismatched_cpu.append(full_x_hat_ukf_m.cpu());␣
↪all_P_hat_ukf_mismatched_cpu.append(full_P_hat_ukf_m.cpu())
        all_x_hat_ukf_ideal_cpu.append(full_x_hat_ukf_i.cpu());␣
↪all_P_hat_ukf_ideal_cpu.append(full_P_hat_ukf_i.cpu())
```

```
        all_x_hat_aekf_mismatched_cpu.append(full_x_hat_aekf_m.cpu());␣
 ↪all_P_hat_aekf_mismatched_cpu.append(full_P_hat_aekf_m.cpu())

        print(f"Dokončena trajektorie {i + 1}/{NUM_TEST_TRAJ}...")

# ==============================================================================
# 5. FINÁLNÍ VÝPOČET A VÝPIS METRIK
# ==============================================================================
# Seznamy pro sběr metrik
mse_bkn, anees_bkn = [], []; mse_classic_knet = []; mse_knetR, anees_knetR =␣
 ↪[], []
mse_ekf_mis, anees_ekf_mis = [], []; mse_ekf_ideal, anees_ekf_ideal = [], []
mse_ukf_mis, anees_ukf_mis = [], []; mse_ukf_ideal, anees_ukf_ideal = [], []
mse_aekf_mis, anees_aekf_mis = [], []

print("\nPočítám finální metriky pro jednotlivé trajektorie...")

with torch.no_grad():
    for i in range(NUM_TEST_TRAJ):
        x_true = all_x_true_cpu[i]
        def get_metrics(x_hat, P_hat):
            mse = F.mse_loss(x_hat[1:], x_true[1:]).item()
            anees = utils.calculate_anees_vectorized(x_true.unsqueeze(0), x_hat.
 ↪unsqueeze(0), P_hat.unsqueeze(0))
            return mse, anees

        # Výpočty pro všechny modely
        mse, anees = get_metrics(all_x_hat_bkn_cpu[i], all_P_hat_bkn_cpu[i]);␣
 ↪mse_bkn.append(mse); anees_bkn.append(anees)
        mse = F.mse_loss(all_x_hat_classic_knet_cpu[i][1:], x_true[1:]).item();␣
 ↪mse_classic_knet.append(mse)
        mse, anees = get_metrics(all_x_hat_knetR_cpu[i],␣
 ↪all_P_hat_knetR_cpu[i]); mse_knetR.append(mse); anees_knetR.append(anees)
        mse, anees = get_metrics(all_x_hat_ekf_mismatched_cpu[i],␣
 ↪all_P_hat_ekf_mismatched_cpu[i]); mse_ekf_mis.append(mse); anees_ekf_mis.
 ↪append(anees)
        mse, anees = get_metrics(all_x_hat_ekf_ideal_cpu[i],␣
 ↪all_P_hat_ekf_ideal_cpu[i]); mse_ekf_ideal.append(mse); anees_ekf_ideal.
 ↪append(anees)
        mse, anees = get_metrics(all_x_hat_ukf_mismatched_cpu[i],␣
 ↪all_P_hat_ukf_mismatched_cpu[i]); mse_ukf_mis.append(mse); anees_ukf_mis.
 ↪append(anees)
        mse, anees = get_metrics(all_x_hat_ukf_ideal_cpu[i],␣
 ↪all_P_hat_ukf_ideal_cpu[i]); mse_ukf_ideal.append(mse); anees_ukf_ideal.
 ↪append(anees)
```

```
        mse, anees = get_metrics(all_x_hat_aekf_mismatched_cpu[i],␣
 ↪all_P_hat_aekf_mismatched_cpu[i]); mse_aekf_mis.append(mse); anees_aekf_mis.
 ↪append(anees)

# Funkce pro bezpečné průměrování
def avg(metric_list): return np.mean([m for m in metric_list if not np.
 ↪isnan(m)])
state_dim_for_nees = all_x_true_cpu[0].shape[1]

# --- Finální výpis tabulky ---
print("\n" + "="*80)
print(f"FINÁLNÍ VÝSLEDKY (průměr přes {NUM_TEST_TRAJ} běhů)")
print("="*80)
print(f"{'Model':<35} | {'Průměrné MSE':<20} | {'Průměrný ANEES':<20}")
print("-" * 80)
print(f"{'--- Data-Driven Models ---':<35} | {'(nižší je lepší)':<20} |␣
 ↪{'(bližší ' + str(float(state_dim_for_nees)) + ' je lepší)':<20}")
print(f"{'Bayesian KNet (BKN)':<35} | {avg(mse_bkn):<20.4f} | {avg(anees_bkn):
 ↪<20.4f}")
print(f"{'KNet (pouze MSE)':<35} | {avg(mse_classic_knet):<20.4f} | {'N/A':
 ↪<20}")
print(f"{'KNet with Known R (KNetR)':<35} | {avg(mse_knetR):<20.4f} |␣
 ↪{avg(anees_knetR):<20.4f}")
print("-" * 80)
print(f"{'--- Model-Based Filters ---':<35} | {'':<20} | {'':<20}")
print(f"{'EKF (Nepřesný model)':<35} | {avg(mse_ekf_mis):<20.4f} |␣
 ↪{avg(anees_ekf_mis):<20.4f}")
print(f"{'UKF (Nepřesný model)':<35} | {avg(mse_ukf_mis):<20.4f} |␣
 ↪{avg(anees_ukf_mis):<20.4f}")
print(f"{'AEKF (Nepřesný model)':<35} | {avg(mse_aekf_mis):<20.4f} |␣
 ↪{avg(anees_aekf_mis):<20.4f}")
print("-" * 80)
print(f"{'--- Benchmarks ---':<35} | {'':<20} | {'':<20}")
print(f"{'EKF (Ideální model)':<35} | {avg(mse_ekf_ideal):<20.4f} |␣
 ↪{avg(anees_ekf_ideal):<20.4f}")
print(f"{'UKF (Ideální model)':<35} | {avg(mse_ukf_ideal):<20.4f} |␣
 ↪{avg(anees_ukf_ideal):<20.4f}")
print("="*80)
```

```
INFO: Všechny natrénované modely nalezeny a přiřazeny.

Generuji 20 testovacích trajektorií o délce 200…
Generování dat dokončeno.
Všechny model-based filtry (EKF, UKF, AEKF) inicializovány.

Vyhodnocuji modely na 20 testovacích trajektoriích…
Dokončena trajektorie 1/20…
```

```
Dokončena trajektorie 2/20…
Dokončena trajektorie 3/20…
Dokončena trajektorie 4/20…
Dokončena trajektorie 5/20…
Dokončena trajektorie 6/20…
Dokončena trajektorie 7/20…
Dokončena trajektorie 8/20…
Dokončena trajektorie 9/20…
Dokončena trajektorie 10/20…
Dokončena trajektorie 11/20…
Dokončena trajektorie 12/20…
Dokončena trajektorie 13/20…
Dokončena trajektorie 14/20…
Dokončena trajektorie 15/20…
Dokončena trajektorie 16/20…
Dokončena trajektorie 17/20…
Dokončena trajektorie 18/20…
Dokončena trajektorie 19/20…
Dokončena trajektorie 20/20…

Počítám finální metriky pro jednotlivé trajektorie…

================================================================================
FINÁLNÍ VÝSLEDKY (průměr přes 20 běhů)
================================================================================
Model                                | Průměrné MSE         | Průměrný ANEES
--------------------------------------------------------------------------------
--- Data-Driven Models ---           | (nižší je lepší)     | (bližší 2.0 je
lepší)
Bayesian KNet (BKN)                  | 0.1038               | 2.9628
KNet (pouze MSE)                     | 0.0842               | N/A
KNet with Known R (KNetR)            | 0.0842               | 2.0341
--------------------------------------------------------------------------------
--- Model-Based Filters ---          |                      |
EKF (Nepřesný model)                 | 0.1415               | 5.0494
UKF (Nepřesný model)                 | 0.1641               | 4.2426
AEKF (Nepřesný model)                | 0.2160               | 8.0220
--------------------------------------------------------------------------------
--- Benchmarks ---                   |                      |
EKF (Ideální model)                  | 0.0838               | 1.9829
UKF (Ideální model)                  | 0.0843               | 0.8396
================================================================================
```