main_runner_nonlinear_system_trajectory_wise

September 26, 2025

[1]:
```
import sys
import os

notebook_path = os.getcwd()
project_root = os.path.dirname(notebook_path)

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]:
```
from models.filter_trajectory_ensemble import trajectory_StateBayesianKalmanNet
from systems.DynamicSystem import DynamicSystem
from training.utils import calculate_anees_vectorized, generate_data
import training.trainer as trainer
```

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

```
Používané zařízení: cuda
```

[5]:
```
import torch
from math import pi

#␣
 ↪==============================================================================
# KROK 1: DEFINICE PARAMETRŮ PRO NELINEÁRNÍ SYSTÉM ("Synthetic")
#␣
 ↪==============================================================================

state_dim_nl = 2
obs_dim_nl = 2
```

```
# --- Parametry reálného systému ("Ground Truth") ---
# Tyto hodnoty odpovídají 'Full' knowledge v jejich kódu
alpha_true = 0.9
beta_true = 1.1
phi_true = 0.1 * pi
delta_true = 0.01
a_true = 1.0
b_true = 1.0
c_true = 0.0

# Definice nelineárních funkcí pomocí lambda
# Důležité: Musí být schopné pracovat s dávkami (batches)!
f_true_nonlinear = lambda x: alpha_true * torch.sin(beta_true * x + phi_true) +␣
 ↪delta_true
h_true_nonlinear = lambda x: a_true * (b_true * x + c_true)**2

# Parametry šumu a počátečních podmínek
Q_true_nl = torch.eye(state_dim_nl) * 0.5  # Šum procesu
R_true_nl = torch.eye(obs_dim_nl) * 0.1  # Šum měření
Ex0_true_nl = torch.tensor([[1.0], [0.0]])
P0_true_nl = torch.eye(state_dim_nl) * 1.5


# --- Parametry modelu systému ("Partial Knowledge") ---
# Tyto hodnoty odpovídají 'Partial' knowledge, simulují nepřesný model
alpha_model = 1.0
beta_model = 1.0
phi_model = 0.0
delta_model = 0.0
a_model = 1.0
b_model = 1.0
c_model = 0.0

# Definice nelineárních funkcí modelu
f_model_nonlinear = lambda x: alpha_model * torch.sin(beta_model * x +␣
 ↪phi_model) + delta_model
h_model_nonlinear = lambda x: a_model * (b_model * x + c_model)**2

# Model může mít i nepřesnou znalost šumu a počátečních podmínek
Q_model_nl = torch.eye(state_dim_nl) * 0.1
R_model_nl = R_true_nl # Předpokládejme, že R známe přesně
Ex0_model_nl = torch.tensor([[0.5], [0.5]])
P0_model_nl = torch.eye(state_dim_nl) * 1.0


#␣
 ↪================================================================================
```

```
# KROK 2: INICIALIZACE OBJEKTŮ SYSTÉMŮ
#␣
 ↪================================================================================
# Ujisti se, že proměnná `device` je definována
# device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

print("\nInicializuji 2D 'Synthetic' nelineární systém...")

# Reálný systém, který bude generovat data
sys_true = DynamicSystem(
    state_dim=state_dim_nl, obs_dim=obs_dim_nl,
    Ex0=Ex0_true_nl, P0=P0_true_nl,
    Q=Q_true_nl, R=R_true_nl,
    f=f_true_nonlinear, h=h_true_nonlinear, # Předáváme funkce, ne matice
    device=device
)

# Model, který bude používat tvůj KalmanNet (s nepřesnými parametry)
sys_model = DynamicSystem(
    state_dim=state_dim_nl, obs_dim=obs_dim_nl,
    Ex0=Ex0_model_nl, P0=P0_model_nl,
    Q=Q_model_nl, R=R_model_nl,
    f=f_model_nonlinear, h=h_model_nonlinear, # Předáváme funkce, ne matice
    device=device
)

print("... Nelineární systém inicializován.")
```

```
Inicializuji 2D 'Synthetic' nelineární systém…
… Nelineární 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 = generate_data(sys_true, num_trajectories=NUM_TRAIN_TRAJ,␣
 ↪seq_len=TRAIN_SEQ_LEN)
x_val, y_val = generate_data(sys_true, num_trajectories=NUM_VALID_TRAJ,␣
 ↪seq_len=VALID_SEQ_LEN)
```

```
x_test, y_test = 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)
```

[11]:
```
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": 800,
    "learning_rate": 1e-4,
    "clip_grad": 10.0,
    "J_samples": 20,
    "validation_period": 20,
    "logging_period": 20,
    "warmup_iterations":200 # 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 = trajectory_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': 800, 'learning_rate': 0.0001,
'clip_grad': 10.0, 'J_samples': 20, 'validation_period': 20, 'logging_period':
20, 'warmup_iterations': 200}
================================================================================
--- Iteration [20/800] ---
    - Total Loss: 0.3804
    - NLL: 0.0000
    - Reg: 0.0026
    - p1=0.602, p2=0.655

--- Validace v iteraci 20 ---
  Průměrný MSE: 0.6219, Průměrný ANEES: 19.8282
  >>> Nové nejlepší VALIDAČNÍ ANEES! Ukládám model. <<<
------------------------------------------------
--- Iteration [40/800] ---
    - Total Loss: 0.8207
    - NLL: 0.0000
    - Reg: 0.0026
    - p1=0.602, p2=0.654

--- Validace v iteraci 40 ---
  Průměrný MSE: 0.6024, Průměrný ANEES: 21.4937
------------------------------------------------
--- Iteration [60/800] ---
    - Total Loss: 0.9874
    - NLL: 0.0000
    - Reg: 0.0026
    - p1=0.602, p2=0.654

--- Validace v iteraci 60 ---
  Průměrný MSE: 0.6026, Průměrný ANEES: 21.3351
------------------------------------------------
--- Iteration [80/800] ---
    - Total Loss: 0.7711
    - NLL: 0.0000
    - Reg: 0.0026
    - p1=0.602, p2=0.654

--- Validace v iteraci 80 ---
  Průměrný MSE: 0.6049, Průměrný ANEES: 19.5311
  >>> Nové nejlepší VALIDAČNÍ ANEES! Ukládám model. <<<
------------------------------------------------
--- Iteration [100/800] ---
    - Total Loss: 1.0453
    - NLL: 0.0000
    - Reg: 0.0026
    - p1=0.602, p2=0.654
```

```
--- Validace v iteraci 100 ---
  Průměrný MSE: 0.6035, Průměrný ANEES: 21.9878
--------------------------------------------------
--- Iteration [120/800] ---
    - Total Loss: 0.4948
    - NLL: 0.0000
    - Reg: 0.0026
    - p1=0.602, p2=0.654

--- Validace v iteraci 120 ---
  Průměrný MSE: 0.5992, Průměrný ANEES: 22.2588
--------------------------------------------------
--- Iteration [140/800] ---
    - Total Loss: 0.7753
    - NLL: 0.0000
    - Reg: 0.0026
    - p1=0.601, p2=0.654

--- Validace v iteraci 140 ---
  Průměrný MSE: 0.6021, Průměrný ANEES: 23.2274
--------------------------------------------------
--- Iteration [160/800] ---
    - Total Loss: 1.1649
    - NLL: 0.0000
    - Reg: 0.0026
    - p1=0.601, p2=0.654

--- Validace v iteraci 160 ---
  Průměrný MSE: 0.5919, Průměrný ANEES: 22.3094
--------------------------------------------------
--- Iteration [180/800] ---
    - Total Loss: 0.3891
    - NLL: 0.0000
    - Reg: 0.0026
    - p1=0.601, p2=0.654

--- Validace v iteraci 180 ---
  Průměrný MSE: 0.5982, Průměrný ANEES: 18.9109
  >>> Nové nejlepší VALIDAČNÍ ANEES! Ukládám model. <<<
--------------------------------------------------
--- Iteration [200/800] ---
    - Total Loss: 0.4479
    - NLL: 0.0000
    - Reg: 0.0026
    - p1=0.601, p2=0.654

--- Validace v iteraci 200 ---
  Průměrný MSE: 0.6023, Průměrný ANEES: 20.1888
```

```
------------------------------------------------
--- Iteration [220/800] ---
    - Total Loss: 4.1163
    - NLL: 4.1136
    - Reg: 0.0026
    - p1=0.602, p2=0.655

--- Validace v iteraci 220 ---
  Průměrný MSE: 0.6695, Průměrný ANEES: 9.6796
  >>> Nové nejlepší VALIDAČNÍ ANEES! Ukládám model. <<<
------------------------------------------------
--- Iteration [240/800] ---
    - Total Loss: 3.9190
    - NLL: 3.9163
    - Reg: 0.0026
    - p1=0.602, p2=0.655

--- Validace v iteraci 240 ---
  Průměrný MSE: 0.7934, Průměrný ANEES: 6.3586
  >>> Nové nejlepší VALIDAČNÍ ANEES! Ukládám model. <<<
------------------------------------------------
--- Iteration [260/800] ---
    - Total Loss: 4.7431
    - NLL: 4.7404
    - Reg: 0.0026
    - p1=0.603, p2=0.656

--- Validace v iteraci 260 ---
  Průměrný MSE: 0.7662, Průměrný ANEES: 5.4870
  >>> Nové nejlepší VALIDAČNÍ ANEES! Ukládám model. <<<
------------------------------------------------
--- Iteration [280/800] ---
    - Total Loss: 2.5458
    - NLL: 2.5431
    - Reg: 0.0026
    - p1=0.603, p2=0.657

--- Validace v iteraci 280 ---
  Průměrný MSE: 0.7566, Průměrný ANEES: 4.9197
  >>> Nové nejlepší VALIDAČNÍ ANEES! Ukládám model. <<<
------------------------------------------------
--- Iteration [300/800] ---
    - Total Loss: 2.8329
    - NLL: 2.8303
    - Reg: 0.0027
    - p1=0.603, p2=0.657

--- Validace v iteraci 300 ---
```

```
  Průměrný MSE: 0.7478, Průměrný ANEES: 4.7359
  >>> Nové nejlepší VALIDAČNÍ ANEES! Ukládám model. <<<
--------------------------------------------------
--- Iteration [320/800] ---
    - Total Loss: 2.7149
    - NLL: 2.7122
    - Reg: 0.0027
    - p1=0.604, p2=0.657

--- Validace v iteraci 320 ---
  Průměrný MSE: 0.7487, Průměrný ANEES: 4.7370
--------------------------------------------------
--- Iteration [340/800] ---
    - Total Loss: 2.7714
    - NLL: 2.7687
    - Reg: 0.0027
    - p1=0.604, p2=0.658

--- Validace v iteraci 340 ---
  Průměrný MSE: 0.7209, Průměrný ANEES: 4.0805
  >>> Nové nejlepší VALIDAČNÍ ANEES! Ukládám model. <<<
--------------------------------------------------
--- Iteration [360/800] ---
    - Total Loss: 7.3170
    - NLL: 7.3144
    - Reg: 0.0027
    - p1=0.604, p2=0.658

--- Validace v iteraci 360 ---
  Průměrný MSE: 0.7048, Průměrný ANEES: 3.9984
  >>> Nové nejlepší VALIDAČNÍ ANEES! Ukládám model. <<<
--------------------------------------------------
--- Iteration [380/800] ---
    - Total Loss: 2.3075
    - NLL: 2.3049
    - Reg: 0.0027
    - p1=0.604, p2=0.658

--- Validace v iteraci 380 ---
  Průměrný MSE: 0.7050, Průměrný ANEES: 3.8841
  >>> Nové nejlepší VALIDAČNÍ ANEES! Ukládám model. <<<
--------------------------------------------------
--- Iteration [400/800] ---
    - Total Loss: 2.0747
    - NLL: 2.0720
    - Reg: 0.0027
    - p1=0.604, p2=0.659
```

```
--- Validace v iteraci 400 ---
  Průměrný MSE: 0.6940, Průměrný ANEES: 3.6679
  >>> Nové nejlepší VALIDAČNÍ ANEES! Ukládám model. <<<
--------------------------------------------------
--- Iteration [420/800] ---
    - Total Loss: 3.8011
    - NLL: 3.7984
    - Reg: 0.0027
    - p1=0.604, p2=0.659

--- Validace v iteraci 420 ---
  Průměrný MSE: 0.7006, Průměrný ANEES: 3.4418
  >>> Nové nejlepší VALIDAČNÍ ANEES! Ukládám model. <<<
--------------------------------------------------
--- Iteration [440/800] ---
    - Total Loss: 4.5363
    - NLL: 4.5336
    - Reg: 0.0027
    - p1=0.604, p2=0.659

--- Validace v iteraci 440 ---
  Průměrný MSE: 0.6739, Průměrný ANEES: 3.3542
  >>> Nové nejlepší VALIDAČNÍ ANEES! Ukládám model. <<<
--------------------------------------------------
--- Iteration [460/800] ---
    - Total Loss: 8.0626
    - NLL: 8.0599
    - Reg: 0.0027
    - p1=0.604, p2=0.659

--- Validace v iteraci 460 ---
  Průměrný MSE: 0.6748, Průměrný ANEES: 3.2163
  >>> Nové nejlepší VALIDAČNÍ ANEES! Ukládám model. <<<
--------------------------------------------------
--- Iteration [480/800] ---
    - Total Loss: 2.5438
    - NLL: 2.5411
    - Reg: 0.0027
    - p1=0.605, p2=0.659

--- Validace v iteraci 480 ---
  Průměrný MSE: 0.6828, Průměrný ANEES: 3.2407
--------------------------------------------------
--- Iteration [500/800] ---
    - Total Loss: 2.4697
    - NLL: 2.4670
    - Reg: 0.0027
    - p1=0.605, p2=0.660
```

```
--- Validace v iteraci 500 ---
  Průměrný MSE: 0.6818, Průměrný ANEES: 3.0227
  >>> Nové nejlepší VALIDAČNÍ ANEES! Ukládám model. <<<
--------------------------------------------------
--- Iteration [520/800] ---
    - Total Loss: 2.8986
    - NLL: 2.8959
    - Reg: 0.0027
    - p1=0.605, p2=0.660

--- Validace v iteraci 520 ---
  Průměrný MSE: 0.6730, Průměrný ANEES: 2.9790
  >>> Nové nejlepší VALIDAČNÍ ANEES! Ukládám model. <<<
--------------------------------------------------
--- Iteration [540/800] ---
    - Total Loss: 2.4036
    - NLL: 2.4010
    - Reg: 0.0027
    - p1=0.605, p2=0.660

--- Validace v iteraci 540 ---
  Průměrný MSE: 0.6778, Průměrný ANEES: 2.9671
  >>> Nové nejlepší VALIDAČNÍ ANEES! Ukládám model. <<<
--------------------------------------------------
--- Iteration [560/800] ---
    - Total Loss: 2.4987
    - NLL: 2.4960
    - Reg: 0.0027
    - p1=0.605, p2=0.660

--- Validace v iteraci 560 ---
  Průměrný MSE: 0.6704, Průměrný ANEES: 2.8342
  >>> Nové nejlepší VALIDAČNÍ ANEES! Ukládám model. <<<
--------------------------------------------------
--- Iteration [580/800] ---
    - Total Loss: 2.0668
    - NLL: 2.0641
    - Reg: 0.0027
    - p1=0.605, p2=0.661

--- Validace v iteraci 580 ---
  Průměrný MSE: 0.6576, Průměrný ANEES: 2.8503
--------------------------------------------------
--- Iteration [600/800] ---
    - Total Loss: 2.4610
    - NLL: 2.4583
    - Reg: 0.0027
```

```
    - p1=0.605, p2=0.661

--- Validace v iteraci 600 ---
  Průměrný MSE: 0.6553, Průměrný ANEES: 2.6961
  >>> Nové nejlepší VALIDAČNÍ ANEES! Ukládám model. <<<
--------------------------------------------------
--- Iteration [620/800] ---
    - Total Loss: 2.1039
    - NLL: 2.1012
    - Reg: 0.0027
    - p1=0.605, p2=0.661

--- Validace v iteraci 620 ---
  Průměrný MSE: 0.6512, Průměrný ANEES: 2.9517
--------------------------------------------------
--- Iteration [640/800] ---
    - Total Loss: 1.9540
    - NLL: 1.9513
    - Reg: 0.0027
    - p1=0.605, p2=0.661

--- Validace v iteraci 640 ---
  Průměrný MSE: 0.6612, Průměrný ANEES: 2.6007
  >>> Nové nejlepší VALIDAČNÍ ANEES! Ukládám model. <<<
--------------------------------------------------
--- Iteration [660/800] ---
    - Total Loss: 2.4527
    - NLL: 2.4500
    - Reg: 0.0027
    - p1=0.605, p2=0.661

--- Validace v iteraci 660 ---
  Průměrný MSE: 0.6457, Průměrný ANEES: 2.6980
--------------------------------------------------
--- Iteration [680/800] ---
    - Total Loss: 2.5783
    - NLL: 2.5756
    - Reg: 0.0027
    - p1=0.605, p2=0.661

--- Validace v iteraci 680 ---
  Průměrný MSE: 0.6557, Průměrný ANEES: 2.6299
--------------------------------------------------
--- Iteration [700/800] ---
    - Total Loss: 3.2628
    - NLL: 3.2601
    - Reg: 0.0027
    - p1=0.605, p2=0.661
```

```
--- Validace v iteraci 700 ---
  Průměrný MSE: 0.6536, Průměrný ANEES: 2.4908
  >>> Nové nejlepší VALIDAČNÍ ANEES! Ukládám model. <<<
--------------------------------------------------
--- Iteration [720/800] ---
    - Total Loss: 2.1175
    - NLL: 2.1149
    - Reg: 0.0027
    - p1=0.605, p2=0.661

--- Validace v iteraci 720 ---
  Průměrný MSE: 0.6547, Průměrný ANEES: 2.4595
  >>> Nové nejlepší VALIDAČNÍ ANEES! Ukládám model. <<<
--------------------------------------------------
--- Iteration [740/800] ---
    - Total Loss: 2.8249
    - NLL: 2.8222
    - Reg: 0.0027
    - p1=0.605, p2=0.662

--- Validace v iteraci 740 ---
  Průměrný MSE: 0.6666, Průměrný ANEES: 2.5411
--------------------------------------------------
--- Iteration [760/800] ---
    - Total Loss: 2.1636
    - NLL: 2.1609
    - Reg: 0.0027
    - p1=0.605, p2=0.662

--- Validace v iteraci 760 ---
  Průměrný MSE: 0.6471, Průměrný ANEES: 2.4096
  >>> Nové nejlepší VALIDAČNÍ ANEES! Ukládám model. <<<
--------------------------------------------------
--- Iteration [780/800] ---
    - Total Loss: 2.0852
    - NLL: 2.0825
    - Reg: 0.0027
    - p1=0.605, p2=0.662

--- Validace v iteraci 780 ---
  Průměrný MSE: 0.6625, Průměrný ANEES: 2.4622
--------------------------------------------------
--- Iteration [800/800] ---
    - Total Loss: 3.1839
    - NLL: 3.1812
    - Reg: 0.0027
    - p1=0.605, p2=0.662
```

```
--- Validace v iteraci 800 ---
  Průměrný MSE: 0.6507, Průměrný ANEES: 2.5548
------------------------------------------------

Trénování dokončeno.
Načítám nejlepší model z iterace 760 s ANEES 2.4096

================================================================================
TRÉNINK DOKONČEN - FINÁLNÍ VÝSLEDKY Z NEJLEPŠÍHO MODELU
================================================================================
Nejlepší model byl nalezen v iteraci: 760
Nejlepší dosažený validační ANEES: 2.4096
--- Metriky odpovídající tomuto nejlepšímu modelu ---
  MSE na validační sadě:       0.6471
  NLL na validační sadě:       0.0000
================================================================================
```

[16]:
```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

model_config = {
    "hidden_size_multiplier": 14,
    "output_layer_multiplier": 4,
    "num_gru_layers": 2,
    "init_min_dropout": 0.5,
    "init_max_dropout": 0.8
}

train_config = {
    "total_train_iter": 800,
    "learning_rate": 1e-4,
    "clip_grad": 10.0,
    "J_samples": 20,
    "validation_period": 20,
    "logging_period": 20,
    "warmup_iterations":800 # 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_knet = trajectory_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_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_knet = 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}")
```

```python
print("="*80)

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

```
================================================================================
Spouštím jeden plnohodnotný tréninkový běh…
Parametry modelu: {'hidden_size_multiplier': 14, 'output_layer_multiplier': 4,
'num_gru_layers': 2, 'init_min_dropout': 0.5, 'init_max_dropout': 0.8}
Parametry tréninku: {'total_train_iter': 800, 'learning_rate': 0.0001,
'clip_grad': 10.0, 'J_samples': 20, 'validation_period': 20, 'logging_period':
20, 'warmup_iterations': 800}
================================================================================
--- Iteration [20/800] ---
    - Total Loss: 0.8070
    - NLL: 0.0000
    - Reg: 0.0068
    - p1=0.781, p2=0.702

--- Validace v iteraci 20 ---
  Průměrný MSE: 1.0987, Průměrný ANEES: 109.3162
  >>> Nové nejlepší VALIDAČNÍ ANEES! Ukládám model. <<<
------------------------------------------------
--- Iteration [40/800] ---
    - Total Loss: 0.7414
    - NLL: 0.0000
    - Reg: 0.0068
    - p1=0.781, p2=0.702

--- Validace v iteraci 40 ---
  Průměrný MSE: 0.8120, Průměrný ANEES: 57.3545
  >>> Nové nejlepší VALIDAČNÍ ANEES! Ukládám model. <<<
------------------------------------------------
--- Iteration [60/800] ---
    - Total Loss: 0.6443
    - NLL: 0.0000
    - Reg: 0.0068
    - p1=0.781, p2=0.702

--- Validace v iteraci 60 ---
  Průměrný MSE: 0.7156, Průměrný ANEES: 46.5186
  >>> Nové nejlepší VALIDAČNÍ ANEES! Ukládám model. <<<
------------------------------------------------
--- Iteration [80/800] ---
    - Total Loss: 0.5780
    - NLL: 0.0000
    - Reg: 0.0068
```

```
    - p1=0.781, p2=0.702

--- Validace v iteraci 80 ---
  Průměrný MSE: 0.6761, Průměrný ANEES: 47.6067
--------------------------------------------------
--- Iteration [100/800] ---
    - Total Loss: 0.4540
    - NLL: 0.0000
    - Reg: 0.0067
    - p1=0.780, p2=0.702

--- Validace v iteraci 100 ---
  Průměrný MSE: 0.6703, Průměrný ANEES: 44.7287
  >>> Nové nejlepší VALIDAČNÍ ANEES! Ukládám model. <<<
--------------------------------------------------
--- Iteration [120/800] ---
    - Total Loss: 1.1564
    - NLL: 0.0000
    - Reg: 0.0067
    - p1=0.780, p2=0.702

--- Validace v iteraci 120 ---
  Průměrný MSE: 0.6645, Průměrný ANEES: 44.3084
  >>> Nové nejlepší VALIDAČNÍ ANEES! Ukládám model. <<<
--------------------------------------------------
--- Iteration [140/800] ---
    - Total Loss: 0.6181
    - NLL: 0.0000
    - Reg: 0.0067
    - p1=0.780, p2=0.703

--- Validace v iteraci 140 ---
  Průměrný MSE: 0.6567, Průměrný ANEES: 43.0867
  >>> Nové nejlepší VALIDAČNÍ ANEES! Ukládám model. <<<
--------------------------------------------------
--- Iteration [160/800] ---
    - Total Loss: 0.8340
    - NLL: 0.0000
    - Reg: 0.0067
    - p1=0.779, p2=0.702

--- Validace v iteraci 160 ---
  Průměrný MSE: 0.6507, Průměrný ANEES: 37.5689
  >>> Nové nejlepší VALIDAČNÍ ANEES! Ukládám model. <<<
--------------------------------------------------
--- Iteration [180/800] ---
    - Total Loss: 0.6689
    - NLL: 0.0000
```

```
    - Reg: 0.0067
    - p1=0.779, p2=0.702

--- Validace v iteraci 180 ---
  Průměrný MSE: 0.6445, Průměrný ANEES: 36.1104
  >>> Nové nejlepší VALIDAČNÍ ANEES! Ukládám model. <<<
--------------------------------------------------
--- Iteration [200/800] ---
    - Total Loss: 0.8622
    - NLL: 0.0000
    - Reg: 0.0067
    - p1=0.779, p2=0.702

--- Validace v iteraci 200 ---
  Průměrný MSE: 0.6397, Průměrný ANEES: 33.5902
  >>> Nové nejlepší VALIDAČNÍ ANEES! Ukládám model. <<<
--------------------------------------------------
--- Iteration [220/800] ---
    - Total Loss: 0.9974
    - NLL: 0.0000
    - Reg: 0.0067
    - p1=0.779, p2=0.702

--- Validace v iteraci 220 ---
  Průměrný MSE: 0.6401, Průměrný ANEES: 30.6007
  >>> Nové nejlepší VALIDAČNÍ ANEES! Ukládám model. <<<
--------------------------------------------------
--- Iteration [240/800] ---
    - Total Loss: 0.5408
    - NLL: 0.0000
    - Reg: 0.0066
    - p1=0.778, p2=0.702

--- Validace v iteraci 240 ---
  Průměrný MSE: 0.6353, Průměrný ANEES: 30.2614
  >>> Nové nejlepší VALIDAČNÍ ANEES! Ukládám model. <<<
--------------------------------------------------
--- Iteration [260/800] ---
    - Total Loss: 0.5530
    - NLL: 0.0000
    - Reg: 0.0066
    - p1=0.778, p2=0.702

--- Validace v iteraci 260 ---
  Průměrný MSE: 0.6357, Průměrný ANEES: 30.8787
--------------------------------------------------
--- Iteration [280/800] ---
    - Total Loss: 0.5959
```

```
    - NLL: 0.0000
    - Reg: 0.0066
    - p1=0.778, p2=0.702

--- Validace v iteraci 280 ---
  Průměrný MSE: 0.6389, Průměrný ANEES: 29.3516
  >>> Nové nejlepší VALIDAČNÍ ANEES! Ukládám model. <<<
--------------------------------------------------
--- Iteration [300/800] ---
    - Total Loss: 0.3894
    - NLL: 0.0000
    - Reg: 0.0066
    - p1=0.777, p2=0.702

--- Validace v iteraci 300 ---
  Průměrný MSE: 0.6389, Průměrný ANEES: 26.7970
  >>> Nové nejlepší VALIDAČNÍ ANEES! Ukládám model. <<<
--------------------------------------------------
--- Iteration [320/800] ---
    - Total Loss: 0.6670
    - NLL: 0.0000
    - Reg: 0.0066
    - p1=0.777, p2=0.702

--- Validace v iteraci 320 ---
  Průměrný MSE: 0.6395, Průměrný ANEES: 27.2774
--------------------------------------------------
--- Iteration [340/800] ---
    - Total Loss: 0.3334
    - NLL: 0.0000
    - Reg: 0.0066
    - p1=0.777, p2=0.702

--- Validace v iteraci 340 ---
  Průměrný MSE: 0.6375, Průměrný ANEES: 25.8185
  >>> Nové nejlepší VALIDAČNÍ ANEES! Ukládám model. <<<
--------------------------------------------------
--- Iteration [360/800] ---
    - Total Loss: 0.6253
    - NLL: 0.0000
    - Reg: 0.0065
    - p1=0.777, p2=0.702

--- Validace v iteraci 360 ---
  Průměrný MSE: 0.6365, Průměrný ANEES: 27.4491
--------------------------------------------------
--- Iteration [380/800] ---
    - Total Loss: 0.6858
```

```
    - NLL: 0.0000
    - Reg: 0.0065
    - p1=0.776, p2=0.702

--- Validace v iteraci 380 ---
  Průměrný MSE: 0.6391, Průměrný ANEES: 26.3771
--------------------------------------------------
--- Iteration [400/800] ---
    - Total Loss: 0.7932
    - NLL: 0.0000
    - Reg: 0.0065
    - p1=0.776, p2=0.702

--- Validace v iteraci 400 ---
  Průměrný MSE: 0.6360, Průměrný ANEES: 27.0970
--------------------------------------------------
--- Iteration [420/800] ---
    - Total Loss: 0.4629
    - NLL: 0.0000
    - Reg: 0.0065
    - p1=0.776, p2=0.702

--- Validace v iteraci 420 ---
  Průměrný MSE: 0.6396, Průměrný ANEES: 25.9186
--------------------------------------------------
--- Iteration [440/800] ---
    - Total Loss: 0.4052
    - NLL: 0.0000
    - Reg: 0.0065
    - p1=0.776, p2=0.702

--- Validace v iteraci 440 ---
  Průměrný MSE: 0.6387, Průměrný ANEES: 26.1910
--------------------------------------------------
--- Iteration [460/800] ---
    - Total Loss: 0.4262
    - NLL: 0.0000
    - Reg: 0.0065
    - p1=0.775, p2=0.702

--- Validace v iteraci 460 ---
  Průměrný MSE: 0.6281, Průměrný ANEES: 25.9542
--------------------------------------------------
--- Iteration [480/800] ---
    - Total Loss: 0.5287
    - NLL: 0.0000
    - Reg: 0.0064
    - p1=0.775, p2=0.702
```

```
--- Validace v iteraci 480 ---
  Průměrný MSE: 0.6366, Průměrný ANEES: 27.9391
--------------------------------------------------
--- Iteration [500/800] ---
    - Total Loss: 0.5806
    - NLL: 0.0000
    - Reg: 0.0064
    - p1=0.775, p2=0.701

--- Validace v iteraci 500 ---
  Průměrný MSE: 0.6322, Průměrný ANEES: 26.5911
--------------------------------------------------
--- Iteration [520/800] ---
    - Total Loss: 0.6290
    - NLL: 0.0000
    - Reg: 0.0064
    - p1=0.775, p2=0.701

--- Validace v iteraci 520 ---
  Průměrný MSE: 0.6356, Průměrný ANEES: 27.3342
--------------------------------------------------
--- Iteration [540/800] ---
    - Total Loss: 0.5752
    - NLL: 0.0000
    - Reg: 0.0064
    - p1=0.774, p2=0.701

--- Validace v iteraci 540 ---
  Průměrný MSE: 0.6365, Průměrný ANEES: 25.9622
--------------------------------------------------
--- Iteration [560/800] ---
    - Total Loss: 0.7391
    - NLL: 0.0000
    - Reg: 0.0064
    - p1=0.774, p2=0.701

--- Validace v iteraci 560 ---
  Průměrný MSE: 0.6303, Průměrný ANEES: 25.4890
  >>> Nové nejlepší VALIDAČNÍ ANEES! Ukládám model. <<<
--------------------------------------------------
--- Iteration [580/800] ---
    - Total Loss: 0.3211
    - NLL: 0.0000
    - Reg: 0.0064
    - p1=0.774, p2=0.701

--- Validace v iteraci 580 ---
```

```
  Průměrný MSE: 0.6425, Průměrný ANEES: 25.9353
--------------------------------------------------
--- Iteration [600/800] ---
    - Total Loss: 0.6587
    - NLL: 0.0000
    - Reg: 0.0063
    - p1=0.773, p2=0.701

--- Validace v iteraci 600 ---
  Průměrný MSE: 0.6322, Průměrný ANEES: 28.3917
--------------------------------------------------
--- Iteration [620/800] ---
    - Total Loss: 0.5361
    - NLL: 0.0000
    - Reg: 0.0063
    - p1=0.773, p2=0.701

--- Validace v iteraci 620 ---
  Průměrný MSE: 0.6334, Průměrný ANEES: 27.5219
--------------------------------------------------
--- Iteration [640/800] ---
    - Total Loss: 0.6456
    - NLL: 0.0000
    - Reg: 0.0063
    - p1=0.773, p2=0.701

--- Validace v iteraci 640 ---
  Průměrný MSE: 0.6379, Průměrný ANEES: 25.8535
--------------------------------------------------
--- Iteration [660/800] ---
    - Total Loss: 0.7487
    - NLL: 0.0000
    - Reg: 0.0063
    - p1=0.772, p2=0.701

--- Validace v iteraci 660 ---
  Průměrný MSE: 0.6355, Průměrný ANEES: 28.6712
--------------------------------------------------
--- Iteration [680/800] ---
    - Total Loss: 0.7737
    - NLL: 0.0000
    - Reg: 0.0063
    - p1=0.772, p2=0.701

--- Validace v iteraci 680 ---
  Průměrný MSE: 0.6354, Průměrný ANEES: 28.0308
--------------------------------------------------
--- Iteration [700/800] ---
```

```
    - Total Loss: 0.3457
    - NLL: 0.0000
    - Reg: 0.0063
    - p1=0.772, p2=0.701

--- Validace v iteraci 700 ---
  Průměrný MSE: 0.6262, Průměrný ANEES: 27.6371
--------------------------------------------------
--- Iteration [720/800] ---
    - Total Loss: 0.8783
    - NLL: 0.0000
    - Reg: 0.0063
    - p1=0.772, p2=0.701

--- Validace v iteraci 720 ---
  Průměrný MSE: 0.6307, Průměrný ANEES: 26.8961
--------------------------------------------------
--- Iteration [740/800] ---
    - Total Loss: 0.4382
    - NLL: 0.0000
    - Reg: 0.0062
    - p1=0.771, p2=0.701

--- Validace v iteraci 740 ---
  Průměrný MSE: 0.6314, Průměrný ANEES: 28.1691
--------------------------------------------------
--- Iteration [760/800] ---
    - Total Loss: 0.5697
    - NLL: 0.0000
    - Reg: 0.0062
    - p1=0.771, p2=0.701

--- Validace v iteraci 760 ---
  Průměrný MSE: 0.6337, Průměrný ANEES: 29.2011
--------------------------------------------------
--- Iteration [780/800] ---
    - Total Loss: 0.5598
    - NLL: 0.0000
    - Reg: 0.0062
    - p1=0.771, p2=0.700

--- Validace v iteraci 780 ---
  Průměrný MSE: 0.6283, Průměrný ANEES: 28.8535
--------------------------------------------------
--- Iteration [800/800] ---
    - Total Loss: 0.6065
    - NLL: 0.0000
    - Reg: 0.0062
```

```
    - p1=0.770, p2=0.700

--- Validace v iteraci 800 ---
  Průměrný MSE: 0.6319, Průměrný ANEES: 29.7355
------------------------------------------------

Trénování dokončeno.
Načítám nejlepší model z iterace 560 s ANEES 25.4890

================================================================================
TRÉNINK DOKONČEN - FINÁLNÍ VÝSLEDKY Z NEJLEPŠÍHO MODELU
================================================================================
Nejlepší model byl nalezen v iteraci: 560
Nejlepší dosažený validační ANEES: 25.4890
--- Metriky odpovídající tomuto nejlepšímu modelu ---
  MSE na validační sadě:       0.6303
  NLL na validační sadě:       0.0000
================================================================================
```

[9]:
```python
from models.StateKalmanNet import StateKalmanNet
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=150,
    lr=1e-4,
    early_stopping_patience=50
)
```

```
/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 [5/150], Train Loss: 0.576674, Val Loss: 0.615347
Epoch [10/150], Train Loss: 0.563771, Val Loss: 0.595021
Epoch [15/150], Train Loss: 0.551133, Val Loss: 0.594988
Epoch [20/150], Train Loss: 0.541918, Val Loss: 0.598352
Epoch [25/150], Train Loss: 0.538681, Val Loss: 0.596736
Epoch [30/150], Train Loss: 0.539116, Val Loss: 0.597162
Epoch [35/150], Train Loss: 0.536365, Val Loss: 0.595077
Epoch [40/150], Train Loss: 0.535018, Val Loss: 0.595874
Epoch [45/150], Train Loss: 0.535097, Val Loss: 0.596208
Epoch [50/150], Train Loss: 0.535883, Val Loss: 0.596279
Epoch [55/150], Train Loss: 0.535896, Val Loss: 0.596214
Epoch [60/150], Train Loss: 0.534199, Val Loss: 0.596216

Early stopping spuštěno po 64 epochách.
Trénování dokončeno.
Načítám nejlepší model s validační chybou: 0.592000
```

[9]:
```
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)
  )
)
```

[15]:
```python
import torch
import torch.nn.functional as F
import numpy as np
from torch.utils.data import TensorDataset, DataLoader

# ================================================================================
# 0. PŘEDPOKLADY
# ================================================================================
# Předpokládá se, že v proměnných níže máte již natrénované modely:
# trained_model_bkn      <-- Váš BayesianKalmanNet (z dvoufázového tréninku)
# trained_model_classic  <-- Váš StateKalmanNet (z tréninku čistě na MSE)
#
# Přejmenujme je pro srozumitelnost v tomto skriptu:
trained_model_bkn = trained_model # Upravte podle názvu vaší proměnné
trained_model_classic = state_knet       # Upravte podle názvu vaší proměnné


# ================================================================================
# 1. KONFIGURACE TESTU
# ================================================================================
```

```
TEST_SEQ_LEN = 100
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 = 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 FILTRŮ PRO POROVNÁNÍ
# ================================================================================
ekf_mismatched = ExtendedKalmanFilter(sys_model)
ekf_ideal = ExtendedKalmanFilter(sys_true)
print("Extended Kalman Filtry inicializovány.")

# ================================================================================
# 4. VYHODNOCOVACÍ SMYČKA
# ================================================================================
all_x_true_cpu, all_x_hat_bkn_cpu, all_P_hat_bkn_cpu = [], [], []
all_x_hat_classic_knet_cpu = [] # Klasický KNet neprodukuje P
all_x_hat_ekf_mismatched_cpu, all_P_hat_ekf_mismatched_cpu = [], []
all_x_hat_ekf_ideal_cpu, all_P_hat_ekf_ideal_cpu = [], []

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

# Důležité: Přepneme oba modely do evaluačního režimu
trained_model_bkn.eval()
trained_model_classic.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. Vyhodnocení 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):
                y_t = y_test_seq_gpu[t, :].unsqueeze(0)
                x_filtered_t, _ = trained_model_bkn.step(y_t)
                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)
        outer_products = diff.unsqueeze(-1) @ diff.unsqueeze(-2)
        covariances_bkn = outer_products.mean(dim=0)

        full_x_hat_bkn = torch.cat([initial_state, predictions_bkn], dim=0)
        P0_val = trained_model_bkn.system_model.P0.to(device)
        full_P_hat_bkn = torch.cat([P0_val.unsqueeze(0), covariances_bkn],␣
↪dim=0)

        # --- B. Vyhodnocení klasického StateKalmanNet ---
        trained_model_classic.reset(batch_size=1, initial_state=initial_state)
        classic_knet_preds = []
        for t in range(1, TEST_SEQ_LEN):
            y_t = y_test_seq_gpu[t, :].unsqueeze(0)
            x_filtered_t = trained_model_classic.step(y_t)
            classic_knet_preds.append(x_filtered_t)

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

        # --- C. Vyhodnocení EKF s nepřesným modelem ---
        Ex0_mismatched = sys_model.Ex0.to(device)
        P0_mismatched = sys_model.P0.to(device)
        ekf_mismatched_results = ekf_mismatched.
↪process_sequence(y_test_seq_gpu, Ex0=Ex0_mismatched, P0=P0_mismatched)
        full_x_hat_ekf_mismatched = torch.cat([Ex0_mismatched.reshape(1, -1),␣
↪ekf_mismatched_results['x_filtered']], dim=0)
        full_P_hat_ekf_mismatched = torch.cat([P0_mismatched.unsqueeze(0),␣
↪ekf_mismatched_results['P_filtered']], dim=0)

        # --- D. Vyhodnocení EKF s perfektním modelem (Benchmark) ---
        Ex0_ideal = sys_true.Ex0.to(device)
        P0_ideal = sys_true.P0.to(device)
        ekf_ideal_results = ekf_ideal.process_sequence(y_test_seq_gpu,␣
↪Ex0=Ex0_ideal, P0=P0_ideal)
        full_x_hat_ekf_ideal = torch.cat([Ex0_ideal.reshape(1, -1),␣
↪ekf_ideal_results['x_filtered']], dim=0)
```

```
        full_P_hat_ekf_ideal = torch.cat([P0_ideal.unsqueeze(0),␣
↪ekf_ideal_results['P_filtered']], dim=0)

        # --- E. Uložení všech výsledků na CPU ---
        min_len = min(full_x_hat_bkn.shape[0], full_x_hat_classic_knet.
↪shape[0], full_x_hat_ekf_mismatched.shape[0], x_true_seq_gpu.shape[0])

        all_x_true_cpu.append(x_true_seq_gpu[:min_len].cpu())
        all_x_hat_bkn_cpu.append(full_x_hat_bkn[:min_len].cpu())
        all_P_hat_bkn_cpu.append(full_P_hat_bkn[:min_len].cpu())
        all_x_hat_classic_knet_cpu.append(full_x_hat_classic_knet[:min_len].
↪cpu())
        all_x_hat_ekf_mismatched_cpu.append(full_x_hat_ekf_mismatched[:min_len].
↪cpu())
        all_P_hat_ekf_mismatched_cpu.append(full_P_hat_ekf_mismatched[:min_len].
↪cpu())
        all_x_hat_ekf_ideal_cpu.append(full_x_hat_ekf_ideal[:min_len].cpu())
        all_P_hat_ekf_ideal_cpu.append(full_P_hat_ekf_ideal[:min_len].cpu())

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

# ==============================================================================
# 5. FINÁLNÍ VÝPOČET A VÝPIS METRIK
# ==============================================================================
mse_bkn_list, anees_bkn_list = [], []
mse_classic_knet_list = [] # ANEES zde není
mse_ekf_mismatched_list, anees_ekf_mismatched_list = [], []
mse_ekf_ideal_list, anees_ekf_ideal_list = [], []

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]

        # Metriky pro Bayesian KalmanNet
        x_hat_bkn = all_x_hat_bkn_cpu[i]
        P_hat_bkn = all_P_hat_bkn_cpu[i]
        mse_bkn_run = F.mse_loss(x_hat_bkn[1:], x_true[1:]).item()
        anees_bkn_run = calculate_anees_vectorized(x_true.unsqueeze(0),␣
↪x_hat_bkn.unsqueeze(0), P_hat_bkn.unsqueeze(0))
        mse_bkn_list.append(mse_bkn_run)
        anees_bkn_list.append(anees_bkn_run)

        # Metriky pro klasický KalmanNet
        x_hat_classic_knet = all_x_hat_classic_knet_cpu[i]
```

```
        mse_classic_knet_run = F.mse_loss(x_hat_classic_knet[1:], x_true[1:]).
↪item()
        mse_classic_knet_list.append(mse_classic_knet_run)

        # Metriky pro nepřesný EKF
        x_hat_ekf_mismatched = all_x_hat_ekf_mismatched_cpu[i]
        P_hat_ekf_mismatched = all_P_hat_ekf_mismatched_cpu[i]
        mse_ekf_mismatched_run = F.mse_loss(x_hat_ekf_mismatched[1:], x_true[1:
↪]).item()
        anees_ekf_mismatched_run = calculate_anees_vectorized(x_true.
↪unsqueeze(0), x_hat_ekf_mismatched.unsqueeze(0), P_hat_ekf_mismatched.
↪unsqueeze(0))
        mse_ekf_mismatched_list.append(mse_ekf_mismatched_run)
        anees_ekf_mismatched_list.append(anees_ekf_mismatched_run)

        # Metriky pro ideální EKF
        x_hat_ekf_ideal = all_x_hat_ekf_ideal_cpu[i]
        P_hat_ekf_ideal = all_P_hat_ekf_ideal_cpu[i]
        mse_ekf_ideal_run = F.mse_loss(x_hat_ekf_ideal[1:], x_true[1:]).item()
        anees_ekf_ideal_run = calculate_anees_vectorized(x_true.unsqueeze(0),␣
↪x_hat_ekf_ideal.unsqueeze(0), P_hat_ekf_ideal.unsqueeze(0))
        mse_ekf_ideal_list.append(mse_ekf_ideal_run)
        anees_ekf_ideal_list.append(anees_ekf_ideal_run)

        print(f"Traj {i+1:2d} | BKN MSE: {mse_bkn_run:.4f} | KNet MSE:␣
↪{mse_classic_knet_run:.4f} | EKF-Mis MSE: {mse_ekf_mismatched_run:.4f} |␣
↪EKF-Ideal MSE: {mse_ekf_ideal_run:.4f}")
        print(f"       | BKN ANEES: {anees_bkn_run:.4f} | KNet ANEES:    N/A    ␣
↪| EKF-Mis ANEES: {anees_ekf_mismatched_run:8.4f} | EKF-Ideal ANEES:␣
↪{anees_ekf_ideal_run:8.4f}")
        print("-" * 110)

# Průměrování výsledků
avg_mse_bkn = np.mean(mse_bkn_list)
avg_anees_bkn = np.mean(anees_bkn_list)
avg_mse_classic_knet = np.mean(mse_classic_knet_list)
avg_mse_ekf_mismatched = np.mean(mse_ekf_mismatched_list)
avg_anees_ekf_mismatched = np.mean(anees_ekf_mismatched_list)
avg_mse_ekf_ideal = np.mean(mse_ekf_ideal_list)
avg_anees_ekf_ideal = np.mean(anees_ekf_ideal_list)
state_dim_for_nees = all_x_true_cpu[0].shape[1]

print("\n" + "="*80)
print(f"FINÁLNÍ VÝSLEDKY (průměr přes {NUM_TEST_TRAJ} běhů)")
print("="*80)
print("\n--- Průměrná MSE (přesnost) ---")
```

```
print(f"Bayesian KNet (BKN):                {avg_mse_bkn:.4f}")
print(f"Klasický KNet (pouze MSE):          {avg_mse_classic_knet:.4f} <--␣
 ↪Nejlepší data-driven přesnost")
print(f"EKF (Nepřesný model):               {avg_mse_ekf_mismatched:.4f}")
print(f"EKF (Ideální model - Benchmark):      {avg_mse_ekf_ideal:.4f} <--␣
 ↪Teoretický limit přesnosti")
print("\n--- Průměrný ANEES (kredibilita/kalibrace) ---")
print(f"Očekávaná hodnota:    {state_dim_for_nees:.4f}")
print("-------------------------------------------------")
print(f"Bayesian KNet (BKN):                {avg_anees_bkn:.4f}")
print(f"Klasický KNet (pouze MSE):          N/A (model neprodukuje kovarianci)")
print(f"EKF (Nepřesný model):               {avg_anees_ekf_mismatched:.4f}")
print(f"EKF (Ideální model - Benchmark):      {avg_anees_ekf_ideal:.4f}")
print("="*80)
```

```
Generuji 20 testovacích trajektorií o délce 100…
Generování dat dokončeno.
Extended Kalman Filtry 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…
Traj  1 | BKN MSE: 0.8754 | KNet MSE: 1.0605 | EKF-Mis MSE: 2.0070 | EKF-Ideal
MSE: 2.6799
       | BKN ANEES: 2.7162 | KNet ANEES:    N/A    | EKF-Mis ANEES: 103.6367 |
EKF-Ideal ANEES: 101.5924
```

```
------------------------------------------------------------------------------
------------------------------
Traj  2 | BKN MSE: 0.6286 | KNet MSE: 0.5704 | EKF-Mis MSE: 1.6215 | EKF-Ideal
MSE: 2.0025
       | BKN ANEES: 2.0019 | KNet ANEES:    N/A    | EKF-Mis ANEES:  75.1305 |
EKF-Ideal ANEES:  51.0218
------------------------------------------------------------------------------
------------------------------
Traj  3 | BKN MSE: 0.6130 | KNet MSE: 0.5308 | EKF-Mis MSE: 1.0916 | EKF-Ideal
MSE: 1.6182
       | BKN ANEES: 1.8800 | KNet ANEES:    N/A    | EKF-Mis ANEES:  46.2399 |
EKF-Ideal ANEES:  57.6847
------------------------------------------------------------------------------
------------------------------
Traj  4 | BKN MSE: 0.6378 | KNet MSE: 0.5272 | EKF-Mis MSE: 1.5862 | EKF-Ideal
MSE: 2.0741
       | BKN ANEES: 1.8587 | KNet ANEES:    N/A    | EKF-Mis ANEES:  85.0592 |
EKF-Ideal ANEES:  84.9973
------------------------------------------------------------------------------
------------------------------
Traj  5 | BKN MSE: 0.7049 | KNet MSE: 0.8093 | EKF-Mis MSE: 1.8217 | EKF-Ideal
MSE: 2.3591
       | BKN ANEES: 2.1851 | KNet ANEES:    N/A    | EKF-Mis ANEES: 100.0731 |
EKF-Ideal ANEES:  83.8721
------------------------------------------------------------------------------
------------------------------
Traj  6 | BKN MSE: 0.7249 | KNet MSE: 0.7251 | EKF-Mis MSE: 2.0973 | EKF-Ideal
MSE: 1.8343
       | BKN ANEES: 2.2305 | KNet ANEES:    N/A    | EKF-Mis ANEES: 103.9281 |
EKF-Ideal ANEES:  79.3101
------------------------------------------------------------------------------
------------------------------
Traj  7 | BKN MSE: 0.6562 | KNet MSE: 0.4812 | EKF-Mis MSE: 2.3487 | EKF-Ideal
MSE: 1.9797
       | BKN ANEES: 2.0712 | KNet ANEES:    N/A    | EKF-Mis ANEES: 120.0514 |
EKF-Ideal ANEES:  67.9487
------------------------------------------------------------------------------
------------------------------
Traj  8 | BKN MSE: 0.6714 | KNet MSE: 0.6611 | EKF-Mis MSE: 1.5188 | EKF-Ideal
MSE: 1.6270
       | BKN ANEES: 2.5629 | KNet ANEES:    N/A    | EKF-Mis ANEES:  87.7809 |
EKF-Ideal ANEES:  71.4247
------------------------------------------------------------------------------
------------------------------
Traj  9 | BKN MSE: 0.6932 | KNet MSE: 0.5986 | EKF-Mis MSE: 2.6493 | EKF-Ideal
MSE: 2.0191
       | BKN ANEES: 2.0661 | KNet ANEES:    N/A    | EKF-Mis ANEES: 157.2696 |
EKF-Ideal ANEES:  70.4671
```

```
--------------------------------------------------------------------------------
------------------------------
Traj 10 | BKN MSE: 0.6596 | KNet MSE: 0.6246 | EKF-Mis MSE: 1.2652 | EKF-Ideal
MSE: 1.7543
       | BKN ANEES: 2.2682 | KNet ANEES:    N/A    | EKF-Mis ANEES:  58.8526 |
EKF-Ideal ANEES:  72.7862
--------------------------------------------------------------------------------
------------------------------
Traj 11 | BKN MSE: 0.8659 | KNet MSE: 0.9562 | EKF-Mis MSE: 2.0590 | EKF-Ideal
MSE: 1.9120
       | BKN ANEES: 2.4813 | KNet ANEES:    N/A    | EKF-Mis ANEES: 106.7832 |
EKF-Ideal ANEES:  73.3330
--------------------------------------------------------------------------------
------------------------------
Traj 12 | BKN MSE: 0.8759 | KNet MSE: 0.7903 | EKF-Mis MSE: 2.8062 | EKF-Ideal
MSE: 4.6073
       | BKN ANEES: 2.0913 | KNet ANEES:    N/A    | EKF-Mis ANEES: 144.6023 |
EKF-Ideal ANEES: 130.0737
--------------------------------------------------------------------------------
------------------------------
Traj 13 | BKN MSE: 0.6121 | KNet MSE: 0.3657 | EKF-Mis MSE: 1.3538 | EKF-Ideal
MSE: 2.6944
       | BKN ANEES: 1.7046 | KNet ANEES:    N/A    | EKF-Mis ANEES:  79.2570 |
EKF-Ideal ANEES:  79.5655
--------------------------------------------------------------------------------
------------------------------
Traj 14 | BKN MSE: 0.8205 | KNet MSE: 0.6971 | EKF-Mis MSE: 3.0866 | EKF-Ideal
MSE: 2.0740
       | BKN ANEES: 2.4253 | KNet ANEES:    N/A    | EKF-Mis ANEES: 188.1530 |
EKF-Ideal ANEES:  82.0892
--------------------------------------------------------------------------------
------------------------------
Traj 15 | BKN MSE: 0.7061 | KNet MSE: 0.5949 | EKF-Mis MSE: 2.5122 | EKF-Ideal
MSE: 2.2083
       | BKN ANEES: 2.2770 | KNet ANEES:    N/A    | EKF-Mis ANEES: 117.1256 |
EKF-Ideal ANEES:  83.3390
--------------------------------------------------------------------------------
------------------------------
Traj 16 | BKN MSE: 0.6096 | KNet MSE: 0.5156 | EKF-Mis MSE: 1.3336 | EKF-Ideal
MSE: 1.6429
       | BKN ANEES: 2.0765 | KNet ANEES:    N/A    | EKF-Mis ANEES:  70.7441 |
EKF-Ideal ANEES:  61.4036
--------------------------------------------------------------------------------
------------------------------
Traj 17 | BKN MSE: 0.7160 | KNet MSE: 0.7032 | EKF-Mis MSE: 1.9633 | EKF-Ideal
MSE: 2.5118
       | BKN ANEES: 2.4209 | KNet ANEES:    N/A    | EKF-Mis ANEES: 100.6792 |
EKF-Ideal ANEES:  87.4269
```

```
------------------------------------------------------------------------------
------------------------------
Traj 18 | BKN MSE: 0.7008 | KNet MSE: 0.6403 | EKF-Mis MSE: 1.7214 | EKF-Ideal
MSE: 1.8221
       | BKN ANEES: 2.7580 | KNet ANEES:    N/A    | EKF-Mis ANEES:  92.0098 |
EKF-Ideal ANEES:  75.2242
------------------------------------------------------------------------------
------------------------------
Traj 19 | BKN MSE: 0.6451 | KNet MSE: 0.6958 | EKF-Mis MSE: 1.4779 | EKF-Ideal
MSE: 1.8567
       | BKN ANEES: 2.0461 | KNet ANEES:    N/A    | EKF-Mis ANEES:  75.4475 |
EKF-Ideal ANEES:  61.9933
------------------------------------------------------------------------------
------------------------------
Traj 20 | BKN MSE: 0.7888 | KNet MSE: 0.8421 | EKF-Mis MSE: 2.3758 | EKF-Ideal
MSE: 2.5471
       | BKN ANEES: 2.3909 | KNet ANEES:    N/A    | EKF-Mis ANEES: 129.1041 |
EKF-Ideal ANEES:  94.8273
------------------------------------------------------------------------------
------------------------------

==============================================================================
FINÁLNÍ VÝSLEDKY (průměr přes 20 běhů)
==============================================================================

--- Průměrná MSE (přesnost) ---
Bayesian KNet (BKN):              0.7103
Klasický KNet (pouze MSE):        0.6695 <-- Nejlepší data-driven přesnost
EKF (Nepřesný model):             1.9349
EKF (Ideální model - Benchmark):    2.1912 <-- Teoretický limit přesnosti

--- Průměrný ANEES (kredibilita/kalibrace) ---
Očekávaná hodnota:   2.0000
------------------------------------------------
Bayesian KNet (BKN):              2.2256
Klasický KNet (pouze MSE):        N/A (model neprodukuje kovarianci)
EKF (Nepřesný model):             102.0964
EKF (Ideální model - Benchmark):    78.5190
==============================================================================
```

[14]:
```python
import torch
from torch.func import jacrev

class ExtendedKalmanFilter:
    def __init__(self, system_model):

        self.device = system_model.Q.device
```

```
        self.f = system_model.f
        self.h = system_model.h

        # Pokud je systém lineární, Jakobiány jsou konstantní matice F a H
        self.is_linear_f = getattr(system_model, 'is_linear_f', False)
        self.is_linear_h = getattr(system_model, 'is_linear_h', False)
        if self.is_linear_f:
            self.F = system_model.F
        if self.is_linear_h:
            self.H = system_model.H


        self.Q = system_model.Q.clone().detach().to(self.device)
        self.R = system_model.R.clone().detach().to(self.device)

        self.state_dim = self.Q.shape[0]
        self.obs_dim = self.R.shape[0]

        self.x_filtered_prev = None
        self.P_filtered_prev = None

        self.reset(system_model.Ex0, system_model.P0)

    def reset(self, Ex0=None, P0=None):
        """
        Inicializuje nebo resetuje stav filtru.
        """
        if Ex0 is not None:
            # Ujistíme se, že má správný tvar [dim, 1]
            self.x_filtered_prev = Ex0.clone().detach().reshape(self.state_dim,␣
↪1)
        if P0 is not None:
            self.P_filtered_prev = P0.clone().detach()

    def predict_step(self, x_filtered, P_filtered):
        if self.is_linear_f:
            F_t = self.F
        else:
            F_t = jacrev(self.f)(x_filtered.squeeze()).reshape(self.state_dim,␣
↪self.state_dim)

        # OPRAVA: Transponujeme vstup pro f() a výstup zpět.
        # [State_Dim, 1] -> .T -> [1, State_Dim] -> f() -> [1, State_Dim] -> .T␣
↪-> [State_Dim, 1]
        x_predict = self.f(x_filtered.T).T
```

```
        P_predict = F_t @ P_filtered @ F_t.T + self.Q
        return x_predict, P_predict

    def update_step(self, x_predict, y_t, P_predict):
        y_t = y_t.reshape(self.obs_dim, 1)

        if self.is_linear_h:
            H_t = self.H
        else:
            H_t = jacrev(self.h)(x_predict.squeeze()).reshape(self.obs_dim,␣
↪self.state_dim)

        # OPRAVA: Transponujeme vstup pro h() a výstup zpět.
        y_predict = self.h(x_predict.T).T
        innovation = y_t - y_predict

        S = H_t @ P_predict @ H_t.T + self.R
        K = P_predict @ H_t.T @ torch.linalg.inv(S)
        x_filtered = x_predict + K @ innovation

        I = torch.eye(self.state_dim, device=self.device)
        P_filtered = (I - K @ H_t) @ P_predict @ (I - K @ H_t).T + K @ self.R @␣
↪K.T

        return x_filtered, P_filtered, K, innovation

    def step(self, y_t):
        """
        Provede jeden kompletní krok filtrace (predict + update) pro online␣
↪použití.
        """
        # 1. Predikce z uloženého interního stavu
        x_predict, P_predict = self.predict_step(self.x_filtered_prev, self.
↪P_filtered_prev)

        # 2. Update s novým měřením
        x_filtered, P_filtered = self.update_step(x_predict, y_t, P_predict)

        # 3. Aktualizace interního stavu pro další volání
        self.x_filtered_prev = x_filtered
        self.P_filtered_prev = P_filtered

        return x_filtered, P_filtered

    def process_sequence(self, y_seq, Ex0=None, P0=None):
            """
```

```
        Zpracuje celou sekvenci měření `y_seq` (offline) a vrátí detailní␣
↪historii.
        """
        # Pokud nejsou zadány, použije defaultní hodnoty z `__init__`
        x_est = Ex0.clone().detach().reshape(self.state_dim, 1) if Ex0 is␣
↪not None else self.x_filtered_prev.clone()
        P_est = P0.clone().detach() if P0 is not None else self.
↪P_filtered_prev.clone()

        seq_len = y_seq.shape[0]

        x_filtered_history = torch.zeros(seq_len, self.state_dim,␣
↪device=self.device)
        P_filtered_history = torch.zeros(seq_len, self.state_dim, self.
↪state_dim, device=self.device)
        x_predict_history = torch.zeros(seq_len, self.state_dim,␣
↪device=self.device)
        P_predict_history = torch.zeros(seq_len, self.state_dim, self.
↪state_dim, device=self.device)
        kalman_gain_history = torch.zeros(seq_len, self.state_dim, self.
↪obs_dim, device=self.device)
        innovation_history = torch.zeros(seq_len, self.obs_dim, device=self.
↪device)

        for t in range(seq_len):
            # 1. Predict
            x_predict, P_predict = self.predict_step(x_est, P_est)

            # 2. Update
            x_est, P_est, K, innovation = self.update_step(x_predict,␣
↪y_seq[t], P_predict)

            x_filtered_history[t] = x_est.squeeze()
            P_filtered_history[t] = P_est
            x_predict_history[t] = x_predict.squeeze()
            P_predict_history[t] = P_predict
            kalman_gain_history[t] = K
            innovation_history[t] = innovation.squeeze()

        return {
            'x_filtered': x_filtered_history,
            'P_filtered': P_filtered_history,
            'x_predict': x_predict_history,
            'P_predict': P_predict_history,
            'Kalman_gain': kalman_gain_history,
            'innovation': innovation_history
```

```
        }
```