TEST_divergence_po_nejake_dobe

November 6, 2025

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


dataset_path = Path('data') / 'data.mat'
if not dataset_path.exists():
    alt = Path.cwd().parent / 'data' / 'data.mat'
    if alt.exists():
        dataset_path = alt
    else:
        raise FileNotFoundError(f"data.mat not found under {Path.cwd()} or its␣
 ↪parent")

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

mat_data = loadmat(dataset_path)
print(mat_data.keys())
```

```
Current notebook path: /home/luky/skola/KalmanNet-for-state-estimation/TAN
Added /home/luky/skola/KalmanNet-for-state-estimation to sys.path
dict_keys(['__header__', '__version__', '__globals__', 'hB', 'souradniceGNSS',
'souradniceX', 'souradniceY', 'souradniceZ'])
```

[2]:
```
import torch
import matplotlib.pyplot as plt
from utils import trainer
from utils import utils
from Systems import DynamicSystem
import Filters
import torch.nn.functional as F
from torch.utils.data import TensorDataset, DataLoader
```

```
import numpy as np
from scipy.io import loadmat
from scipy.interpolate import RegularGridInterpolator
import random

torch.manual_seed(42)
np.random.seed(42)
random.seed(42)
# Pro plnou CUDA reprodukovatelnost (volitelné, ale doporučené)
if torch.cuda.is_available():
    torch.cuda.manual_seed_all(42)
# --------------------

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

```
Používané zařízení: cuda
```

1 4D model

[3]:
```
import torch
from math import pi
from Systems import DynamicSystem # Předpokládám existenci této základní třídy

# --- 1. Definice parametrů (podle PDF a matlab.m) ---
state_dim = 6   # Rozšířený stav: [x, y, vx, vy, ux, uy]
obs_dim = 2     # Měření: [azimut, vzdálenost]
dt = 0.02       # [cite: 11]

# Koeficienty pro neznámý vstup (náhodná procházka)
alpha_Q = 1e-3  # [cite: 21]
alpha_P = 1e-6  # [cite: 21]

# --- 2. Sestavení rozšířené matice přechodu F (F_aug) ---
# Toto je F_aug = [F_base, B; 0, I] z tvého matlab.m
# F_base (Constant Velocity)
F_base = torch.tensor([[1, 0, dt, 0],
                       [0, 1, 0, dt],
                       [0, 0, 1, 0],
                       [0, 0, 0, 1]], dtype=torch.float)
# B matice (jak vstup 'u' ovlivňuje stav)
# Změna rychlosti je 1*u (ne dt*u, podle tvého matlab.m)
B_base = torch.tensor([[0, 0],
                       [0, 0],
                       [1, 0],  # ux ovlivňuje vx
                       [0, 1]], dtype=torch.float) # uy ovlivňuje vy
```

```
# Sestavení finální 6x6 F matice
F_aug = torch.zeros(6, 6, dtype=torch.float)
F_aug[0:4, 0:4] = F_base
F_aug[0:4, 4:6] = B_base
F_aug[4:6, 4:6] = torch.eye(2) # u_{k+1} = I * u_k

# --- 3. Sestavení rozšířené matice šumu Q (Q_aug) ---
# Šum základního stavu
Q_base = torch.eye(4, dtype=torch.float) * 1e-2 # [cite: 11]
# Šum náhodné procházky vstupu
Q_u = torch.eye(2, dtype=torch.float) * alpha_Q # [cite: 19]
# Diagonální sestavení
Q_aug = torch.block_diag(Q_base, Q_u)

# --- 4. Matice šumu měření R ---
R_val = torch.tensor([[4e-4 * (pi/180)**2, 0],  # [cite: 15]
                      [0, 1e-4]], dtype=torch.float) # [cite: 15]

# --- 5. Počáteční podmínky (x_0_aug, P_0_aug) ---
x_0_base = torch.tensor([0, 0, 0, 0], dtype=torch.float) # [cite: 16]
x_0_u = torch.zeros(2, dtype=torch.float) # [cite: 20]
x_0_aug = torch.cat([x_0_base, x_0_u])

P_0_base = torch.eye(4, dtype=torch.float) # [cite: 16]
P_0_u = torch.eye(2, dtype=torch.float) * alpha_P # [cite: 20]
P_0_aug = torch.block_diag(P_0_base, P_0_u)


# --- 6. Definice systémových funkcí (f, h) ---

def f_linear_augmented(x):
    """
    Lineární přechodová funkce f(x_k) = F_aug * x_k
    x: [B, 6]
    """
    # Musíme transponovat pro správné maticové násobení s dávkou
    return (F_aug.to(x.device) @ x.unsqueeze(-1)).squeeze(-1)

def h_polar(x):
    """
    Nelineární pozorovací funkce h(x_k) = [azimut, vzdalenost]
    x: [B, 6] (použije pouze x[:, 0] a x[:, 1])
    """
    # Získání pozic z dávky
    px = x[:, 0]
    py = x[:, 1]
```

```
    eps = 1e-6  # Malá hodnota pro stabilitu
    # Výpočet polárních souřadnic [cite: 13]
    azimuth = torch.atan2(py, px+eps)

    range_val = torch.sqrt(px**2 + py**2)
    # azimuth = torch.tensor(1.0, device=x.device).expand_as(range_val)
    # Vrácení jako [B, 2]
    return torch.stack([azimuth, range_val], dim=1)

def h_linear(x):
    # x je [B, 6]
    # H matice [2, 6] = [[1, 0, 0, 0, 0, 0], [0, 1, 0, 0, 0, 0]]
    return x[:, 0:2]

obs_dim = 2 # Zůstává 2
# R_val = torch.tensor([[1.0, 0.0], [0.0, 1.0]], dtype=torch.float) #
 ↪Zjednodušený šum měření (1m std)


# --- 7. Vytvoření instance DynamicSystem ---
print("Vytvářím instanci systému pro Benchmark (Semestrální práce)...")

# Předpokládám, že tvá třída DynamicSystemTAN dědí z DynamicSystem
# a můžeme ji použít, i když ignorujeme TAN-specifické argumenty.
# Pokud ne, nahraď 'DynamicSystemTAN' za 'DynamicSystem'.
system_model = DynamicSystem(
    state_dim=state_dim,
    obs_dim=obs_dim,
    Q=Q_aug.float(),
    R=R_val.float(),
    Ex0=x_0_aug.float(),
    P0=P_0_aug.float(),
    f=f_linear_augmented,  # <-- Naše nová lineární f()
    h=h_polar,             # <-- Naše nelineární h()
    device=device
)

print(f"Systémový model pro Benchmark (6D stav, 2D měření) vytvořen.")
```

```
Vytvářím instanci systému pro Benchmark (Semestrální práce)…
Systémový model pro Benchmark (6D stav, 2D měření) vytvořen.
```

[4]:
```
# import torch
# from torch.utils.data import TensorDataset, DataLoader
# from copy import deepcopy
# import numpy as np
# import random
```

```
# # Importujeme jen 'trainer' z tvých utils
# from utils import trainer

# # --- Definice pomocné funkce pro generování dat ---
# # (Tato funkce je v pořádku, není třeba ji měnit)
# def generate_data(system_model, num_trajectories, seq_len, base_dist, device):
#     """
#     Generuje data plně vektorizovaným způsobem.
#     """
#     state_dim = system_model.state_dim
#     obs_dim = system_model.obs_dim

#     x_data = torch.zeros(num_trajectories, seq_len, state_dim, device=device)
#     y_data = torch.zeros(num_trajectories, seq_len, obs_dim, device=device)

#     x_k = base_dist.sample((num_trajectories,))
#     y_k = system_model.measure(x_k)

#     x_data[:, 0, :] = x_k
#     y_data[:, 0, :] = y_k

#     with torch.no_grad():
#         for t in range(1, seq_len):
#             x_k = system_model.step(x_k)
#             y_k = system_model.measure(x_k)
#             x_data[:, t, :] = x_k
#             y_data[:, t, :] = y_k

#     return x_data, y_data
# # --- Konec pomocné funkce ---


# # --- Konfigurace ---
# TRAIN_SEQ_LEN = 150
# VALID_SEQ_LEN = 150
# NUM_TRAIN_SETS = 100       # Kolik různých počátečních podmínek
# TRAJ_PER_SET_TRAIN = 4     # Kolik trajektorií na jednu poč. podmínku
# NUM_VALID_SETS = 50
# TRAJ_PER_SET_VALID = 2
# BATCH_SIZE = 256

# device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
# print(f"Používané zařízení: {device}")

# # --- Načtení systémového modelu (6D Benchmark) ---
# original_system_model = system_model
# default_P0 = original_system_model.P0.clone()
```

```
# default_Ex0 = original_system_model.Ex0.clone()
# state_dim = original_system_model.state_dim
# obs_dim = original_system_model.obs_dim

# print(f"Benchmark model načten: State Dim={state_dim}, Obs Dim={obs_dim}")
# print(f"Výchozí Ex0 (střed): {default_Ex0.cpu().numpy()}")

# # --- ZMĚNA: Definice "Meta-Distribuce" pro robustnost ---
# # Místo startu vždy z [1000, 1000], budeme startovní body náhodně
# # posouvat v definovaném rozsahu.
# POS_MEAN_RANGE = 2000.0  # Náhodný posun pozice v rozsahu +/- 2000m
# VEL_MEAN_RANGE = 20.0    # Náhodný posun rychlosti v rozsahu +/- 20m/s

# print(f"Budu generovat sady trajektorií s počáteční pozicí v rozsahu +/-␣
 ↪{POS_MEAN_RANGE} m")
# print(f"a počáteční rychlostí v rozsahu +/- {VEL_MEAN_RANGE} m/s (relativně k␣
 ↪výchozímu Ex0).")


# # --- Generování trénovacích dat (D=200) ---
# print("Generuji trénovací data pro 6D benchmark (s vysokou variabilitou)...")
# all_x_train = []
# all_y_train = []

# for i in range(NUM_TRAIN_SETS):
#     if (i+1) % 10 == 0:
#         print(f"  Generuji trénovací sadu {i+1}/{NUM_TRAIN_SETS}...")

#     # --- ZMĚNA: Vytvoříme NOVÉ náhodné Ex0 pro tuto sadu ---

#     # 1. Vytvoříme náhodný posun (Uniform(-R, R) = (rand*2 - 1)*R)
#     random_pos_offset = (torch.rand(2, device=device) * 2 - 1) *␣
 ↪POS_MEAN_RANGE
#     random_vel_offset = (torch.rand(2, device=device) * 2 - 1) *␣
 ↪VEL_MEAN_RANGE

#     # 2. Sestavíme nový Ex0
#     current_Ex0 = default_Ex0.clone()
#     current_Ex0[0:2] += random_pos_offset # Posun pozic
#     current_Ex0[2:4] += random_vel_offset # Posun rychlostí

#     # 3. Vytvoříme distribuci jen pro tuto sadu trajektorií
#     # P0 (nejistota startu) zůstává stejná
#     current_base_dist = torch.distributions.MultivariateNormal(current_Ex0,␣
 ↪default_P0)

#     x_batch, y_batch = generate_data(
```

```
#         original_system_model, # Můžeme použít originál
#         num_trajectories=TRAJ_PER_SET_TRAIN,
#         seq_len=TRAIN_SEQ_LEN,
#         base_dist=current_base_dist, # <-- Předáme NOVOU distribuci
#         device=device
#     )
#     # --- Konec změny ---

#     all_x_train.append(x_batch)
#     all_y_train.append(y_batch)

# x_train = torch.cat(all_x_train, dim=0)
# y_train = torch.cat(all_y_train, dim=0)
# print(f"Finální trénovací data: x={x_train.shape}, y={y_train.shape}")

# # --- Generování validačních dat (D=400) ---
# print("Generuji validační data pro 6D benchmark (s vysokou variabilitou)...")
# all_x_val = []
# all_y_val = []
# for i in range(NUM_VALID_SETS):
#     if (i+1) % 10 == 0:
#         print(f"  Generuji validační sadu {i+1}/{NUM_VALID_SETS}...")

#     # --- ZMĚNA: Totéž pro validaci ---
#     random_pos_offset = (torch.rand(2, device=device) * 2 - 1) *␣
 ↪POS_MEAN_RANGE
#     random_vel_offset = (torch.rand(2, device=device) * 2 - 1) *␣
 ↪VEL_MEAN_RANGE

#     current_Ex0 = default_Ex0.clone()
#     current_Ex0[0:2] += random_pos_offset
#     current_Ex0[2:4] += random_vel_offset

#     current_base_dist = torch.distributions.MultivariateNormal(current_Ex0,␣
 ↪default_P0)

#     x_batch, y_batch = generate_data(
#         original_system_model,
#         num_trajectories=TRAJ_PER_SET_VALID,
#         seq_len=VALID_SEQ_LEN,
#         base_dist=current_base_dist, # <-- Předáme NOVOU distribuci
#         device=device
#     )
#     # --- Konec změny ---

#     all_x_val.append(x_batch)
#     all_y_val.append(y_batch)
```

```
# x_val = torch.cat(all_x_val, dim=0)
# y_val = torch.cat(all_y_val, dim=0)
# print(f"Finální validační data: x={x_val.shape}, y={y_val.shape}")

# # --- VÝPOČET NORMALIZAČNÍCH STATISTIK ---
# print("\nPočítám normalizační statistiky (průměr a std) z trénovacích dat...")
# x_train_flat = x_train.view(-1, state_dim)
# x_mean = x_train_flat.mean(dim=0).to(device)
# x_std = x_train_flat.std(dim=0).to(device)
# x_std[x_std == 0] = 1.0

# # Tyto hodnoty teď budou vypadat mnohem robustněji:
# print(f"  Vypočtený průměr (x_mean): {x_mean.cpu().numpy()}")
# print(f"  Vypočtená odchylka (x_std): {x_std.cpu().numpy()}")

# # --- Vytvoření DataLoaderů ---
# 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)

# print("\nDataLoadery jsou připraveny pro trénink.")
```

[5]:
```
import torch
from torch.utils.data import TensorDataset, DataLoader
from copy import deepcopy
import numpy as np
import random
from utils import trainer # Importuješ svůj trainer
from state_NN_models import StateKalmanNet_v2 # Importuješ svůj model
from Systems import DynamicSystem  # Importuješ svůj systém
import torch.autograd
torch.autograd.set_detect_anomaly(True)
# --- Zde vlož definici tvé třídy DynamicSystem ---
# (Předpokládám, že je v Systems.py, jak naznačuješ)
# class DynamicSystem:
#     ... (celý tvůj kód pro DynamicSystem) ...

# --- Zde vlož tvůj kód pro definici modelu (f, h, F, Q, R...) ---
# (Předpokládám, že je v Systems.py)
# state_dim = 6
# obs_dim = 2
# ...
# def f_linear_augmented(x): ...
# def h_polar(x): ...
```

```python
# ...
# system_model = DynamicSystem(...)

# --- Pomocná funkce pro generování dat (beze změny) ---
def generate_data(system_model, num_trajectories, seq_len, base_dist, device):
    state_dim = system_model.state_dim
    obs_dim = system_model.obs_dim
    x_data = torch.zeros(num_trajectories, seq_len, state_dim, device=device)
    y_data = torch.zeros(num_trajectories, seq_len, obs_dim, device=device)

    x_k = base_dist.sample((num_trajectories,))

    # Krok 1 (Rada autora): Zkontroluj NaN v h(x) hned na začátku
    y_k = system_model.measure(x_k)
    if torch.any(torch.isnan(y_k)):
        print("VAROVÁNÍ: Detekován NaN v y_k při generování dat (krok 0)!")

    x_data[:, 0, :] = x_k
    y_data[:, 0, :] = y_k

    with torch.no_grad():
        for t in range(1, seq_len):
            x_k = system_model.step(x_k)
            y_k = system_model.measure(x_k)

            # Krok 1 (Rada autora): Průběžná kontrola
            if torch.any(torch.isnan(y_k)):
                print(f"VAROVÁNÍ: Detekován NaN v y_k při generování dat (krok␣
 ↪{t})!")

            x_data[:, t, :] = x_k
            y_data[:, t, :] = y_k

    return x_data, y_data

# --- Konfigurace ---
TRAIN_SEQ_LEN = 100       # D (Délka segmentu pro trénink)
VALID_SEQ_LEN = 200      # D (Délka segmentu pro validaci)
NUM_TRAIN_SETS = 150    # Počet různých startovních pozic (Více dat!)
TRAJ_PER_SET_TRAIN = 10  # Kolik trajektorií na jednu pozici
NUM_VALID_SETS = 100
TRAJ_PER_SET_VALID = 5
BATCH_SIZE = 256

TEST_SEQ_LEN = 800
TRAJ_PER_SET_TEST = 1
```

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

# --- Načtení systémového modelu (6D Benchmark) ---
# (Předpokládá, že 'system_model' je definován v Systems.py a importován)
original_system_model = system_model
default_P0 = original_system_model.P0.clone()
default_Ex0 = original_system_model.Ex0.clone()
state_dim = original_system_model.state_dim
obs_dim = original_system_model.obs_dim
print(f"Benchmark model načten: State Dim={state_dim}, Obs Dim={obs_dim}")

# --- ÚPRAVA: Generování dat mimo nespojitost (atan2) ---
# Budeme generovat trajektorie POUZE v pravé polorovině (x > 0),
# kde je atan2 spojitá.
POS_MEAN_RANGE_X_MIN = 500.0  # Minimální startovní pozice X
POS_MEAN_RANGE_X_MAX = 5000.0 # Maximální startovní pozice X
POS_MEAN_RANGE_Y = 5000.0     # Rozsah Y +/-
VEL_MEAN_RANGE = 20.0         # Rozsah rychlosti +/-

print(f"Generuji trajektorie v 'bezpečné zóně': x > {POS_MEAN_RANGE_X_MIN}")

# --- Generování trénovacích dat ---
print("Generuji trénovací data...")
all_x_train = []
all_y_train = []

for i in range(NUM_TRAIN_SETS):
    # Vytvoříme náhodný posun POUZE v bezpečné zóně
    random_pos_offset_x = (torch.rand((), device=device) *␣
 ↪(POS_MEAN_RANGE_X_MAX - POS_MEAN_RANGE_X_MIN)) + POS_MEAN_RANGE_X_MIN
    random_pos_offset_y = (torch.rand((), device=device) * 2 - 1) *␣
 ↪POS_MEAN_RANGE_Y
    random_vel_offset = (torch.rand(2, device=device) * 2 - 1) * VEL_MEAN_RANGE

    current_Ex0 = default_Ex0.clone()
    current_Ex0[0] += random_pos_offset_x
    current_Ex0[1] += random_pos_offset_y
    current_Ex0[2:4] += random_vel_offset

    current_base_dist = torch.distributions.MultivariateNormal(current_Ex0,␣
 ↪default_P0)

    x_batch, y_batch = generate_data(
        original_system_model,
        num_trajectories=TRAJ_PER_SET_TRAIN,
        seq_len=TRAIN_SEQ_LEN,
```

```
        base_dist=current_base_dist,
        device=device
    )
    all_x_train.append(x_batch)
    all_y_train.append(y_batch)

x_train = torch.cat(all_x_train, dim=0)
y_train = torch.cat(all_y_train, dim=0)
print(f"Finální trénovací data: x={x_train.shape}, y={y_train.shape}")

# --- Generování validačních dat ---
print("Generuji validační data...")
all_x_val = []
all_y_val = []
for i in range(NUM_VALID_SETS):
    # Opět v bezpečné zóně
    random_pos_offset_x = (torch.rand((), device=device) *␣
 ↪(POS_MEAN_RANGE_X_MAX - POS_MEAN_RANGE_X_MIN)) + POS_MEAN_RANGE_X_MIN
    random_pos_offset_y = (torch.rand((), device=device) * 2 - 1) *␣
 ↪POS_MEAN_RANGE_Y
    random_vel_offset = (torch.rand(2, device=device) * 2 - 1) * VEL_MEAN_RANGE

    current_Ex0 = default_Ex0.clone()
    current_Ex0[0] += random_pos_offset_x
    current_Ex0[1] += random_pos_offset_y
    current_Ex0[2:4] += random_vel_offset

    current_base_dist = torch.distributions.MultivariateNormal(current_Ex0,␣
 ↪default_P0)

    x_batch, y_batch = generate_data(
        original_system_model,
        num_trajectories=TRAJ_PER_SET_VALID,
        seq_len=VALID_SEQ_LEN,
        base_dist=current_base_dist,
        device=device
    )
    all_x_val.append(x_batch)
    all_y_val.append(y_batch)

x_val = torch.cat(all_x_val, dim=0)
y_val = torch.cat(all_y_val, dim=0)
print(f"Finální validační data: x={x_val.shape}, y={y_val.shape}")

# --- VÝPOČET NORMALIZAČNÍCH STATISTIK (Rada autora #4) ---
print("\nPočítám normalizační statistiky (průměr a std) z trénovacích dat...")
# Stavy
```

```
x_train_flat = x_train.view(-1, state_dim)
x_mean = x_train_flat.mean(dim=0).to(device)
x_std = x_train_flat.std(dim=0).to(device)
x_std[x_std == 0] = 1.0 # Zabráníme dělení nulou
print(f"  Vypočtený průměr stavů (x_mean): {x_mean.cpu().numpy()}")
print(f"  Vypočtená odchylka stavů (x_std): {x_std.cpu().numpy()}")

# Měření
y_train_flat = y_train.view(-1, obs_dim)
y_mean = y_train_flat.mean(dim=0).to(device)
y_std = y_train_flat.std(dim=0).to(device)
y_std[y_std == 0] = 1.0 # Zabráníme dělení nulou
print(f"  Vypočtený průměr měření (y_mean): {y_mean.cpu().numpy()}")
print(f"  Vypočtená odchylka měření (y_std): {y_std.cpu().numpy()}")



print("Generuji testovací data...")
all_x_test = []
all_y_test = []
for i in range(1): # Můžeme použít stejný počet jako pro validaci
    random_pos_offset_x = (torch.rand((), device=device) *␣
 ↪(POS_MEAN_RANGE_X_MAX - POS_MEAN_RANGE_X_MIN)) + POS_MEAN_RANGE_X_MIN
    random_pos_offset_y = (torch.rand((), device=device) * 2 - 1) *␣
 ↪POS_MEAN_RANGE_Y
    random_vel_offset = (torch.rand(2, device=device) * 2 - 1) * VEL_MEAN_RANGE

    current_Ex0 = default_Ex0.clone()
    current_Ex0[0] += random_pos_offset_x
    current_Ex0[1] += random_pos_offset_y
    current_Ex0[2:4] += random_vel_offset

    current_base_dist = torch.distributions.MultivariateNormal(current_Ex0,␣
 ↪default_P0)

    x_batch, y_batch = generate_data(
        original_system_model,
        num_trajectories=TRAJ_PER_SET_TEST,
        seq_len=TEST_SEQ_LEN, # Použijeme delší sekvenci pro test
        base_dist=current_base_dist,
        device=device
    )
    all_x_test.append(x_batch)
    all_y_test.append(y_batch)

x_test = torch.cat(all_x_test, dim=0)
y_test = torch.cat(all_y_test, dim=0)
```

```
print(f"Finální testovací data: x={x_test.shape}, y={y_test.shape}")

# --- Vytvoření DataLoaderů ---
# Používáme RAW (nenormalizovaná) data. Model provede normalizaci interně.
train_dataset = TensorDataset(x_train, y_train)
val_dataset = TensorDataset(x_val, y_val)
test_dataset = TensorDataset(x_test, y_test)
train_loader = DataLoader(train_dataset, batch_size=BATCH_SIZE, shuffle=True)
val_loader = DataLoader(val_dataset, batch_size=BATCH_SIZE, shuffle=False)
test_loader = DataLoader(test_dataset, batch_size=1, shuffle=False)
print("\nDataLoadery jsou připraveny pro trénink.")
# --- Inicializace modelu (s normalizačními statistikami) ---
print("Inicializuji model StateKalmanNet_v2 s normalizačními statistikami...")
```

```
Používané zařízení: cuda
Benchmark model načten: State Dim=6, Obs Dim=2
Generuji trajektorie v 'bezpečné zóně': x > 500.0
Generuji trénovací data…
Finální trénovací data: x=torch.Size([1500, 100, 6]), y=torch.Size([1500, 100,
2])
Generuji validační data…
Finální validační data: x=torch.Size([500, 200, 6]), y=torch.Size([500, 200, 2])

Počítám normalizační statistiky (průměr a std) z trénovacích dat…
  Vypočtený průměr stavů (x_mean): [ 2.5636851e+03  5.0658745e+01  9.1917914e-01
-1.8693146e-01
  1.9709710e-03 -2.0085992e-03]
  Vypočtená odchylka stavů (x_std): [1.27262097e+03 3.07472192e+03
1.47232065e+01 1.51776152e+01
 2.24774048e-01 2.26145387e-01]
  Vypočtený průměr měření (y_mean): [3.4362879e-02 3.9847346e+03]
  Vypočtená odchylka měření (y_std): [8.7013990e-01 1.3305455e+03]
Generuji testovací data…
Finální testovací data: x=torch.Size([1, 800, 6]), y=torch.Size([1, 800, 2])

DataLoadery jsou připraveny pro trénink.
Inicializuji model StateKalmanNet_v2 s normalizačními statistikami…
```

[7]:
```
state_knet2 = StateKalmanNet_v2(
    system_model=original_system_model,
    device=device,
).to(device)

# --- Spuštění tréninku ---
print("Spouštím trénink...")
trained_model = trainer.train_state_KalmanNet_sliding_window(
    model=state_knet2,
```

```
    train_loader=train_loader,
    val_loader=val_loader,
    device=device,
    epochs=100,
    lr=1e-4, # Vyšší LR, jak radil autor
    clip_grad=1.0,
    early_stopping_patience=20,
    tbptt_k=2,
    tbptt_w=8, # (2, 4, 50) nebo (2, 8, 50)
    optimizer_=torch.optim.Adam,
    weight_decay_=1e-3,

)
```

```
DNN_KalmanNet_v2: max_gain_element set to 2.0
Inicializuji váhy pro StateKalmanNet_v2 (Arch 1)…
Spouštím trénink…
INFO: Detekováno z atributu modelu, že vrací kovarianci: False
INFO: Spouštím trénink s TBPTT(k=2, w=8)
Nové nejlepší model uloženo! Epoch [2/100], Train Loss: inf, Val Loss:
1544177184.000000
Nové nejlepší model uloženo! Epoch [4/100], Train Loss: 20088.674400, Val Loss:
52816.519531
Epoch [5/100], Train Loss: 3726.824195, Val Loss: 425336271016034304.000000
Nové nejlepší model uloženo! Epoch [6/100], Train Loss: 270020.297920, Val Loss:
3251.907471
Nové nejlepší model uloženo! Epoch [8/100], Train Loss: 58.612613, Val Loss:
696.800964
Epoch [10/100], Train Loss: 314.616136, Val Loss: 934792655984.000000
Epoch [15/100], Train Loss: 47.334966, Val Loss: 46588102.375000
Nové nejlepší model uloženo! Epoch [18/100], Train Loss: 290.192278, Val Loss:
681.749695
Epoch [20/100], Train Loss: 94.384333, Val Loss: 2560255.328125
Nové nejlepší model uloženo! Epoch [22/100], Train Loss: 34.320384, Val Loss:
410.610596
Nové nejlepší model uloženo! Epoch [24/100], Train Loss: 2305.617105, Val Loss:
362.998520
Epoch [25/100], Train Loss: 1303.481148, Val Loss: 342.769043
Nové nejlepší model uloženo! Epoch [25/100], Train Loss: 1303.481148, Val Loss:
342.769043
Nové nejlepší model uloženo! Epoch [26/100], Train Loss: 38.808789, Val Loss:
340.303497
Nové nejlepší model uloženo! Epoch [29/100], Train Loss: 32.816998, Val Loss:
332.774216
Epoch [30/100], Train Loss: 44.363599, Val Loss:
2368549903452193526601822565826 56.000000
Nové nejlepší model uloženo! Epoch [31/100], Train Loss: 32009.127301, Val Loss:
```

```
304.836746
Nové nejlepší model uloženo! Epoch [34/100], Train Loss: 31.555678, Val Loss:
290.384659
Epoch [35/100], Train Loss: 29.366835, Val Loss: 282.190430
Nové nejlepší model uloženo! Epoch [35/100], Train Loss: 29.366835, Val Loss:
282.190430
Nové nejlepší model uloženo! Epoch [37/100], Train Loss: 36.256449, Val Loss:
221.663399
Nové nejlepší model uloženo! Epoch [39/100], Train Loss: 79.725705, Val Loss:
220.913307
Epoch [40/100], Train Loss: 27.692673, Val Loss: 192.409203
Nové nejlepší model uloženo! Epoch [40/100], Train Loss: 27.692673, Val Loss:
192.409203
Nové nejlepší model uloženo! Epoch [41/100], Train Loss: 24.724486, Val Loss:
130.840370
Epoch [45/100], Train Loss: 32.379339, Val Loss: 143.203354
Nové nejlepší model uloženo! Epoch [48/100], Train Loss: 22.635819, Val Loss:
80.740555
Epoch [50/100], Train Loss: 184849518616615392.000000, Val Loss: 153.466843
Epoch [55/100], Train Loss: 20.594294, Val Loss: 73.256882
Nové nejlepší model uloženo! Epoch [55/100], Train Loss: 20.594294, Val Loss:
73.256882
Epoch [60/100], Train Loss: 22.742331, Val Loss:
256690542280315228473024514621 44.000000
Epoch [65/100], Train Loss: 42.381432, Val Loss: 89.745129
Nové nejlepší model uloženo! Epoch [69/100], Train Loss: 13.935620, Val Loss:
61.749096
Epoch [70/100], Train Loss: 11.961183, Val Loss: 56.215866
Nové nejlepší model uloženo! Epoch [70/100], Train Loss: 11.961183, Val Loss:
56.215866
Nové nejlepší model uloženo! Epoch [71/100], Train Loss: 10.756314, Val Loss:
51.602747
Nové nejlepší model uloženo! Epoch [73/100], Train Loss: 919223342.637703, Val
Loss: 48.764713
Nové nejlepší model uloženo! Epoch [74/100], Train Loss: 11.359499, Val Loss:
47.961052
Epoch [75/100], Train Loss: 12.585319, Val Loss: 59.171585
Nové nejlepší model uloženo! Epoch [77/100], Train Loss: 16.683546, Val Loss:
47.270226
Nové nejlepší model uloženo! Epoch [79/100], Train Loss: 23.560531, Val Loss:
43.840952
Epoch [80/100], Train Loss: 10.645045, Val Loss: 51.502171
Epoch [85/100], Train Loss: 14.875047, Val Loss: 133.539518
Epoch [90/100], Train Loss: 150.069004, Val Loss: 347.859688
Epoch [95/100], Train Loss: 10.171741, Val Loss: 74.521114

Early stopping spuštěno po 99 epochách.
Trénování dokončeno.
```

```
Načítám nejlepší model s validační chybou: 43.840952
```

[ ]:
```
# import torch
# import torch.nn as nn
# from torch.utils.data import TensorDataset, DataLoader
# import numpy as np
# import os
# import random
# from copy import deepcopy
# from state_NN_models import KalmanNet_Arch1
# from utils import trainer
# # ----------------------------

# # Nastavení seedu
# torch.manual_seed(42)
# np.random.seed(42)
# random.seed(42)

# # IMPLEMENTACE BEZ TANH A BEZ OMEZENÍ STAVŮ
# # Vytvoření modelu (KalmanNet_Arch1) s hyperparametry autorů
# state_knet2 = KalmanNet_Arch1(
#     system_model,
#     device=device,
#     hidden_size_multiplier=10,
#     output_layer_multiplier=4,
#     gru_layers=1

# ).to(device)

# print(state_knet2)

# trainer.train_state_KalmanNet_sliding_window(
#     model=state_knet2,
#     train_loader=train_loader,
#     val_loader=val_loader,
#     device=device,
#     epochs=200,                  # Podle kódu autorů
#     lr=1e-3,                    # Podle kódu autorů (0.001)
#     early_stopping_patience=30, # Můžeme nechat
#     clip_grad=1.0,              # Podle kódu autorů (gradient_clip_val=1)
#     tbptt_k=2,                  # Podle kódu autorů (detach_step=2)
#     tbptt_w=8                   # Podle kódu autorů (slide_win_size=4 nebo 8)
# )
```

[ ]:
```
# import torch
# import torch.nn as nn
# from torch.utils.data import TensorDataset, DataLoader
```

```
# import numpy as np
# import os
# import random
# from copy import deepcopy
# from state_NN_models import KalmanNet_Arch2
# from utils import trainer
# # ----------------------------

# # Nastavení seedu
# torch.manual_seed(42)
# np.random.seed(42)
# random.seed(42)

# # IMPLEMENTACE BEZ TANH A BEZ OMEZENÍ STAVŮ
# # Vytvoření modelu (KalmanNet_Arch2) s hyperparametry autorů
# state_knet2 = KalmanNet_Arch2(
#     system_model,
#     device=device,
#     in_mult_KNet=5,     # Podle kódu autorů
#     out_mult_KNet=40    # Podle kódu autorů
# ).to(device)

# print(state_knet2)

# trainer.train_state_KalmanNet_sliding_window(
#     model=state_knet2,
#     train_loader=train_loader,
#     val_loader=val_loader,
#     device=device,
#     epochs=200,                  # Podle kódu autorů
#     lr=1e-4,                    # Podle kódu autorů (0.001)
#     early_stopping_patience=20, # Můžeme nechat
#     clip_grad=1.0,              # Podle kódu autorů (gradient_clip_val=1)
#     tbptt_k=2,                  # Podle kódu autorů (detach_step=2)
#     tbptt_w=8                   # Podle kódu autorů (slide_win_size=4 nebo 8)
# )
```

[ ]:
```
# import torch
# import torch.nn as nn
# from torch.utils.data import TensorDataset, DataLoader
# import numpy as np
# import os
# import random
# from copy import deepcopy
# from state_NN_models import KalmanNet_Arch2_tanh
# from utils import trainer
# # ----------------------------
```

```
# # Nastavení seedu
# torch.manual_seed(42)
# np.random.seed(42)
# random.seed(42)

# # IMPLEMENTACE S TANH PRO OMEZENÍ K A CLAMP STAVŮ
# # Vytvoření modelu (KalmanNet_Arch2_tanh) s hyperparametry autorů
# state_knet3 = KalmanNet_Arch2_tanh(
#     system_model,
#     device=device,
#     in_mult_KNet=5,     # Podle kódu autorů
#     out_mult_KNet=40    # Podle kódu autorů
# ).to(device)

# print(state_knet3)

# trainer.train_state_KalmanNet_sliding_window_statistiky(
#     model=state_knet3,
#     train_loader=train_loader,
#     val_loader=val_loader,
#     device=device,
#     x_mean=x_mean,           # PŘIDÁNO: průměr pro normalizaci
#     x_std=x_std,             # PŘIDÁNO: std pro normalizaci
#     epochs=200,                  # Podle kódu autorů
#     lr=1e-4,                    # Podle kódu autorů (0.001)
#     early_stopping_patience=20, # Můžeme nechat
#     clip_grad=1.0,              # Podle kódu autorů (gradient_clip_val=1)
#     tbptt_k=2,                  # Podle kódu autorů (detach_step=2)
#     tbptt_w=8                   # Podle kódu autorů (slide_win_size=4 nebo 8)
# )
```

[ ]:
```
# import torch
# import torch.nn as nn
# from torch.utils.data import TensorDataset, DataLoader
# import numpy as np
# import os
# import random
# from copy import deepcopy
# from state_NN_models import KalmanNet_Arch2_tanh
# from utils import trainer
# # ----------------------------

# # Nastavení seedu
# torch.manual_seed(42)
# np.random.seed(42)
# random.seed(42)
```

```
# # IMPLEMENTACE S TANH PRO OMEZENÍ K A CLAMP STAVŮ
# # Vytvoření modelu (KalmanNet_Arch2_tanh) s hyperparametry autorů
# state_knet2 = KalmanNet_Arch2_tanh(
#     system_model,
#     device=device,
#     in_mult_KNet=5,     # Podle kódu autorů
#     out_mult_KNet=40    # Podle kódu autorů
# ).to(device)

# print(state_knet2)

# trainer.train_state_KalmanNet_sliding_window_weighted_mse(
#     model=state_knet2,
#     train_loader=train_loader,
#     val_loader=val_loader,
#     device=device,              # PŘIDÁNO: std pro normalizaci
#     epochs=200,                   # Podle kódu autorů
#     lr=1e-3,                     # Podle kódu autorů (0.001)
#     early_stopping_patience=30, # Můžeme nechat
#     clip_grad=1.0,               # Podle kódu autorů (gradient_clip_val=1)
#     tbptt_k=2,                   # Podle kódu autorů (detach_step=2)
#     tbptt_w=8,                    # Podle kódu autorů (slide_win_size=4 nebo␣
 ↪8)
#     pos_weight=1.0,
#     vel_weight=1000.0
# )
```

[ ]:
```
# 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
# from state_NN_models import StateKalmanNet_v2_4D_tan

# # Nastavení seedu pro reprodukovatelnost tohoto běhu
# torch.manual_seed(42)
# np.random.seed(42)
# random.seed(42)
# state_knet = StateKalmanNet_v2_4D_tan(system_model=system_model,␣
 ↪device=device, hidden_size_multiplier=14, output_layer_multiplier=4,␣
 ↪num_gru_layers=2).to(device)
```

```
# print(state_knet)
# trainer.train_state_KalmanNet(
#     model=state_knet,
#     train_loader=train_loader,
#     val_loader=val_loader,
#     device=device,
#     epochs=200,
#     lr=1e-5,
#     early_stopping_patience=20,
#     clip_grad=1.0
# )
```

[8]:
```
import torch
import torch.nn.functional as F
import numpy as np
from torch.utils.data import TensorDataset, DataLoader



# ================================================================================
# 1. KONFIGURACE TESTU
# ================================================================================
# TEST_SEQ_LEN = 800 # Změňte zpět na 100 nebo kolik potřebujete
# NUM_TEST_TRAJ = 1
# J_SAMPLES_TEST = 25

# #␣
 ↪==============================================================================
# # 2. PŘÍPRAVA DAT (OPRAVENO)
# #␣
 ↪==============================================================================
# print(f"\nGeneruji {NUM_TEST_TRAJ} testovacích trajektorií o délce␣
 ↪{TEST_SEQ_LEN}...")

# # Nyní předáme oříznutou sekvenci 'u', takže i 'x' a 'y' budou mít správnou␣
 ↪délku.
# x_test, y_test = utils.generate_data(
#     system_model,
#     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.")

# print("y shape:", y_test.shape) # Mělo by být [1, 20, 1]
```

```
# print("x shape:", x_test.shape) # Mělo by být [1, 20, 3]



# ================================================================================
# 3. INICIALIZACE FILTRŮ
# ================================================================================
ukf_ideal = Filters.UnscentedKalmanFilter(system_model)
pf_sir_ideal = Filters.ParticleFilter(system_model, num_particles=10000)

# ================================================================================
# 4. VYHODNOCOVACÍ SMYČKA (OPRAVENO)
# ================================================================================
all_x_true_cpu = []
all_x_hat_ukf_ideal_cpu, all_P_hat_ukf_ideal_cpu = [], []
all_x_hat_pf_sir_ideal_cpu, all_P_hat_pf_sir_ideal_cpu = [], []
all_x_hat_classic_knet_cpu = []
all_x_hat_bkn_cpu, all_P_hat_bkn_cpu = [], []
all_x_hat_knet_F3_cpu = []
all_x_hat_classic_knet2_cpu = []
# all_x_hat_classic_knet3_cpu = []
NUM_TEST_TRAJ = 1
all_knet2_diagnostics_cpu = []
# all_knet3_diagnostics_cpu = []
print(f"\nVyhodnocuji modely na {NUM_TEST_TRAJ} testovacích trajektoriích...")

# state_knet.eval()
state_knet2.eval()
# state_knet3.eval()
# state_knet_F3.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)
        initial_state_cpu = initial_state.squeeze(0).cpu()

        # # --- A. Bayesian KalmanNet (Trajectory-wise) ---
        # ensemble_trajectories = []
        # for j in range(J_SAMPLES_TEST):
        #     state_bkn_knet.reset(batch_size=1, initial_state=initial_state)
        #     current_x_hats = []
        #     for t in range(1, TEST_SEQ_LEN):
        #         x_filtered_t, _ = state_bkn_knet.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([system_model.P0.unsqueeze(0),␣
↪covariances_bkn], dim=0)


        # --- B. Klasický StateKalmanNet (pouze MSE) ---
        # state_knet.reset(batch_size=1, initial_state=initial_state)
        # classic_knet_preds = []
        # for t in range(1, TEST_SEQ_LEN):
        #     x_filtered_t = state_knet.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)
        # diagnostics = state_knet.get_diagnostics()
        # all_knet_diagnostics_cpu.append(diagnostics)

        state_knet2.reset(batch_size=1, initial_state=initial_state)
        classic_knet2_preds = []
        for t in range(1, TEST_SEQ_LEN):
            x_filtered_t = state_knet2.step(y_test_seq_gpu[t, :].unsqueeze(0))
            classic_knet2_preds.append(x_filtered_t)
        full_x_hat_classic_knet2 = torch.cat([initial_state, torch.
↪cat(classic_knet2_preds, dim=0)], dim=0)
        # diagnostics_knet2 = state_knet2.get_diagnostics()

        # state_knet3.reset(batch_size=1, initial_state=initial_state)
        # classic_knet3_preds = []
        # for t in range(1, TEST_SEQ_LEN):
        #     x_filtered_t = state_knet3.step(y_test_seq_gpu[t, :].unsqueeze(0))
        #     classic_knet3_preds.append(x_filtered_t)
        # full_x_hat_classic_knet3 = torch.cat([initial_state, torch.
↪cat(classic_knet3_preds, dim=0)], dim=0)
        # diagnostics_knet3 = state_knet3.get_diagnostics()

        ukf_i_res = ukf_ideal.process_sequence(
            y_seq=y_test_seq_gpu,
             Ex0=initial_state,
            P0=system_model.P0
        )
        full_x_hat_ukf_i = ukf_i_res['x_filtered']
```

```
        full_P_hat_ukf_i = ukf_i_res['P_filtered']

        pf_sir_i_res = pf_sir_ideal.process_sequence(y_test_seq_gpu,␣
↪Ex0=initial_state,P0=system_model.P0)
        full_x_hat_pf_sir_i = pf_sir_i_res['x_filtered']
        full_P_hat_pf_sir_i = pf_sir_i_res['P_filtered']
        full_particles_history_pf_sir_i = pf_sir_i_res['particles_history']
        print(f"PF-SIR (ideální model) dokončen pro trajektorii {i + 1}/
↪{NUM_TEST_TRAJ}.")

        all_x_true_cpu.append(x_true_seq_gpu.cpu())
        # all_x_hat_knet_F3_cpu.append(full_x_hat_classic_knet_F3.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_knet2_cpu.append(full_x_hat_classic_knet2.cpu())
        # all_x_hat_classic_knet3_cpu.append(full_x_hat_classic_knet3.cpu())
        # all_x_hat_classic_knet_cpu.append(full_x_hat_classic_knet.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_pf_sir_ideal_cpu.append(full_x_hat_pf_sir_i.cpu());␣
↪all_P_hat_pf_sir_ideal_cpu.append(full_P_hat_pf_sir_i.cpu())

        # all_knet2_diagnostics_cpu.append(diagnostics_knet2)
        # all_knet3_diagnostics_cpu.append(diagnostics_knet3)
        print(f"Dokončena trajektorie {i + 1}/{NUM_TEST_TRAJ}...")

# ==============================================================================
# 5. FINÁLNÍ VÝPOČET A VÝPIS METRIK
# ==============================================================================
# Seznamy pro sběr metrik
mse_bkn, anees_bkn = [], []; mse_ukf_ideal, anees_ukf_ideal = [], []
# mse_classic_knet = []

mse_pf_sir_ideal, anees_pf_sir_ideal = [], []
mse_classic_knet3 = []
mse_classic_knet2 = []
# mse_knet_F3 = []

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

with torch.no_grad():
    for i in range(1):
        x_true = all_x_true_cpu[i]
        def get_metrics(x_hat_full, P_hat_full):
            if x_hat_full.shape[0] != x_true.shape[0] or P_hat_full.shape[0] !=␣
↪x_true.shape[0]:
```

```
                raise ValueError(f"Nesoulad délek! x_true: {x_true.shape[0]},␣
↪x_hat: {x_hat_full.shape[0]}, P_hat: {P_hat_full.shape[0]}")

            # Porovnáváme od kroku t=1
            mse = F.mse_loss(x_hat_full[1:], x_true[1:]).item()
            # ANEES se také typicky počítá od t=1 (ignoruje počáteční nejistotu␣
↪P0)
            anees = utils.calculate_anees_vectorized(
                x_true[1:].unsqueeze(0),
                x_hat_full[1:].unsqueeze(0),
                P_hat_full[1:].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_knet_F3_cpu[i][1:], x_true[1:]).item();␣
↪mse_knet_F3.append(mse)
        mse = F.mse_loss(all_x_hat_classic_knet2_cpu[i][1:], x_true[1:]).item();
↪ mse_classic_knet2.append(mse)
        # mse = F.mse_loss(all_x_hat_classic_knet3_cpu[i][1:], x_true[1:]).
↪item(); mse_classic_knet3.append(mse)
        # 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_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_pf_sir_ideal_cpu[i],␣
↪all_P_hat_pf_sir_ideal_cpu[i]); mse_pf_sir_ideal.append(mse);␣
↪anees_pf_sir_ideal.append(anees)
        print("\n" + "="*80)
        print(f"trajektorie: {i + 1}/{NUM_TEST_TRAJ}")
        print("="*80)
        print("-" * 80)
        # print(f"{'Bayesian KNet (BKN)':<35} | {(mse_bkn[i]):<20.4f} |␣
↪{(anees_bkn[i]):<20.4f}")
        # print(f"{'KNet F3 (pouze MSE)':<35} | {(mse_knet_F3[i]):<20.4f} | {'N/
↪A':<20}")
        print(f"{'KNet2 (pouze MSE)':<35} | {(mse_classic_knet2[i]):<20.4f} |␣
↪{'N/A':<20}")
        # print(f"{'KNet3 (pouze MSE)':<35} | {(mse_classic_knet3[i]):<20.4f} |␣
↪{'N/A':<20}")
        # print(f"{'KNet (pouze MSE)':<35} | {(mse_classic_knet[i]):<20.4f} |␣
↪{'N/A':<20}")
```

```
        print(f"{'UKF (Ideální model)':<35} | {(mse_ukf_ideal[i]):<20.4f} |␣
↪{(anees_ukf_ideal[i]):<20.4f}")
        print(f"{'PF-SIR (Ideální model)':<35} | {(mse_pf_sir_ideal[i]):<20.4f}␣
↪| {(anees_pf_sir_ideal[i]):<20.4f}")
        print("="*80)

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("-" * 80)
print(f"{'--- Model-Based Filters ---':<35} | {'':<20} | {'':<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"{'KNet2 (pouze MSE)':<35} | {avg(mse_classic_knet2):<20.4f} | {'N/A':
↪<20}")
# print(f"{'KNet3 (pouze MSE)':<35} | {avg(mse_classic_knet3):<20.4f} | {'N/A':
↪<20}")
# print(f"{'KNet F3 (pouze MSE)':<35} | {avg(mse_knet_F3):<20.4f} | {'N/A':
↪<20}")
print("-" * 80)
print(f"{'--- Benchmarks ---':<35} | {'':<20} | {'':<20}")
print(f"{'UKF (Ideální model)':<35} | {avg(mse_ukf_ideal):<20.4f} |␣
↪{avg(anees_ukf_ideal):<20.4f}")
print(f"{'PF-SIR (Ideální model)':<35} | {avg(mse_pf_sir_ideal):<20.4f} |␣
↪{avg(anees_pf_sir_ideal):<20.4f}")
print("="*80)
```

```
Vyhodnocuji modely na 1 testovacích trajektoriích…
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
```

```
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
```

```
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
```

```
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
```

```
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
```

```
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
```

```
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
```

```
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
```

```
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
Varování: Součet vah je téměř nulový. Resetuji na uniformní rozdělení.
PF-SIR (ideální model) dokončen pro trajektorii 1/1.
Dokončena trajektorie 1/1…

Počítám finální metriky pro jednotlivé trajektorie…

================================================================================
```

```
trajektorie: 1/1
================================================================================
--------------------------------------------------------------------------------
KNet2 (pouze MSE)                    | 27387.0664          | N/A
UKF (Ideální model)                  | 1.3622              | 5.8581
PF-SIR (Ideální model)               | 21758.9453          | 411.3665
================================================================================

================================================================================
FINÁLNÍ VÝSLEDKY (průměr přes 1 běhů)
================================================================================
Model                                | Průměrné MSE        | Průměrný ANEES
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--- Model-Based Filters ---          |                     |
KNet2 (pouze MSE)                    | 27387.0664          | N/A
--------------------------------------------------------------------------------
--- Benchmarks ---                   |                     |
UKF (Ideální model)                  | 1.3622              | 5.8581
PF-SIR (Ideální model)               | 21758.9453          | 411.3665
================================================================================
```

[9]:
```python
import matplotlib.pyplot as plt
import numpy as np
import torch
import torch.nn.functional as F

# --- Zvolte trajektorii k analýze ---
index = 0
if index < 0: index = 0

# ================================================================================
# 1. PŘÍPRAVA DAT
# ================================================================================
print("Připravuji data pro vykreslení...")

# --- Načtení dat trajektorií ---
x_true_tensor = all_x_true_cpu[index]
x_pf_tensor = all_x_hat_pf_sir_ideal_cpu[index]
x_knet2_tensor = all_x_hat_classic_knet2_cpu[index] # Model bez Tanh
# x_knet3_tensor = all_x_hat_classic_knet3_cpu[index] # Model s Tanh

x_true_plot = x_true_tensor.numpy()
num_steps = x_true_plot.shape[0]
time_axis = np.arange(num_steps)
gain_time_axis = np.arange(1, num_steps) # Zisky a inovace jsou od t=1
```

```
# --- Výpočet RMSE pro oba modely ---
rmse_knet2_per_step = torch.sqrt((x_knet2_tensor - x_true_tensor)**2).numpy()
# rmse_knet3_per_step = torch.sqrt((x_knet3_tensor - x_true_tensor)**2).numpy()

# --- Pomocná funkce pro extrakci diagnostiky ---
def extract_diagnostics(diagnostics_dict, time_axis, gain_time_axis):
    diag_data = {}
    plot_flags = {'gains': False, 'h_norm': False, 'innov_norm': False,␣
 ↪'inputs_norm': False}

    try:
        # --- 1. Kalmanovy zisky (K_t) ---
        # K má tvar [T-1, B=1, state_dim=6, obs_dim=2]
        kalman_gains_history = diagnostics_dict['K_history']

        # Reakce na 1. měření (Azimut)
        gains_col0_cpu = [K[0, :, 0].cpu().numpy() for K in␣
 ↪kalman_gains_history]
        diag_data['gains_col0_np'] = np.array(gains_col0_cpu) # [T-1, 6]

        # Reakce na 2. měření (Vzdálenost)
        gains_col1_cpu = [K[0, :, 1].cpu().numpy() for K in␣
 ↪kalman_gains_history]
        diag_data['gains_col1_np'] = np.array(gains_col1_cpu) # [T-1, 6]

        if diag_data['gains_col0_np'].shape[0] == len(gain_time_axis):
            plot_flags['gains'] = True
        else:
            print(f"Varování: Délka historie zisku K␣
 ↪({diag_data['gains_col0_np'].shape[0]}) neodpovídá časové ose␣
 ↪({len(gain_time_axis)}).")

        # --- 2. Skryté stavy (h_t) ---
        # h_history je list slovníků (délka T)
        h_history = diagnostics_dict['h_history']
        # Spočítáme L2 normu všech tří GRU stavů sečteně
        h_norms = [
            (torch.norm(h['h_Q'].squeeze(1)) +
             torch.norm(h['h_Sigma'].squeeze(1)) +
             torch.norm(h['h_S'].squeeze(1))).item()
            for h in h_history
        ]
        diag_data['h_norms_np'] = np.array(h_norms) # [T]

        if len(diag_data['h_norms_np']) == len(time_axis):
            plot_flags['h_norm'] = True
        else:
```

```
            print(f"Varování: Délka historie stavu h␣
↪({len(diag_data['h_norms_np'])}) neodpovídá časové ose ({len(time_axis)}).")

        # --- 3. Inovace (Delta y_t) ---
        # innovation_history je list tenzorů (délka T-1)
        innovation_history = diagnostics_dict['innovation_history']
        innov_norms = [torch.norm(innov.squeeze(0)).item() for innov in␣
↪innovation_history]
        diag_data['innov_norms_np'] = np.array(innov_norms) # [T-1]

        if len(diag_data['innov_norms_np']) == len(gain_time_axis):
            plot_flags['innov_norm'] = True
        else:
            print(f"Varování: Délka historie inovace␣
↪({len(diag_data['innov_norms_np'])}) neodpovídá časové ose␣
↪({len(gain_time_axis)}).")

        # --- 4. Vstupní Rysy (F1..F4) ---
        # inputs_history je list tenzorů (délka T-1)
        inputs_history = diagnostics_dict['inputs_history']
        inputs_norms = [torch.norm(inp.squeeze(0)).item() for inp in␣
↪inputs_history]
        diag_data['inputs_norms_np'] = np.array(inputs_norms) # [T-1]

        if len(diag_data['inputs_norms_np']) == len(gain_time_axis):
            plot_flags['inputs_norm'] = True
        else:
            print(f"Varování: Délka historie vstupů␣
↪({len(diag_data['inputs_norms_np'])}) neodpovídá časové ose␣
↪({len(gain_time_axis)}).")

    except Exception as e:
        print(f"Nastala chyba při extrakci diagnostiky: {e}")

    return diag_data, plot_flags

# --- Extrakce diagnostiky pro oba modely ---
try:
    knet2_diag_data, knet2_plot_flags =␣
↪extract_diagnostics(all_knet2_diagnostics_cpu[index], time_axis,␣
↪gain_time_axis)
    plot_diag_knet2 = True
except (NameError, IndexError):
    print("Varování: Diagnostika pro 'KNet2' nenalezena. Grafy nebudou␣
↪vykresleny.")
    plot_diag_knet2 = False
```

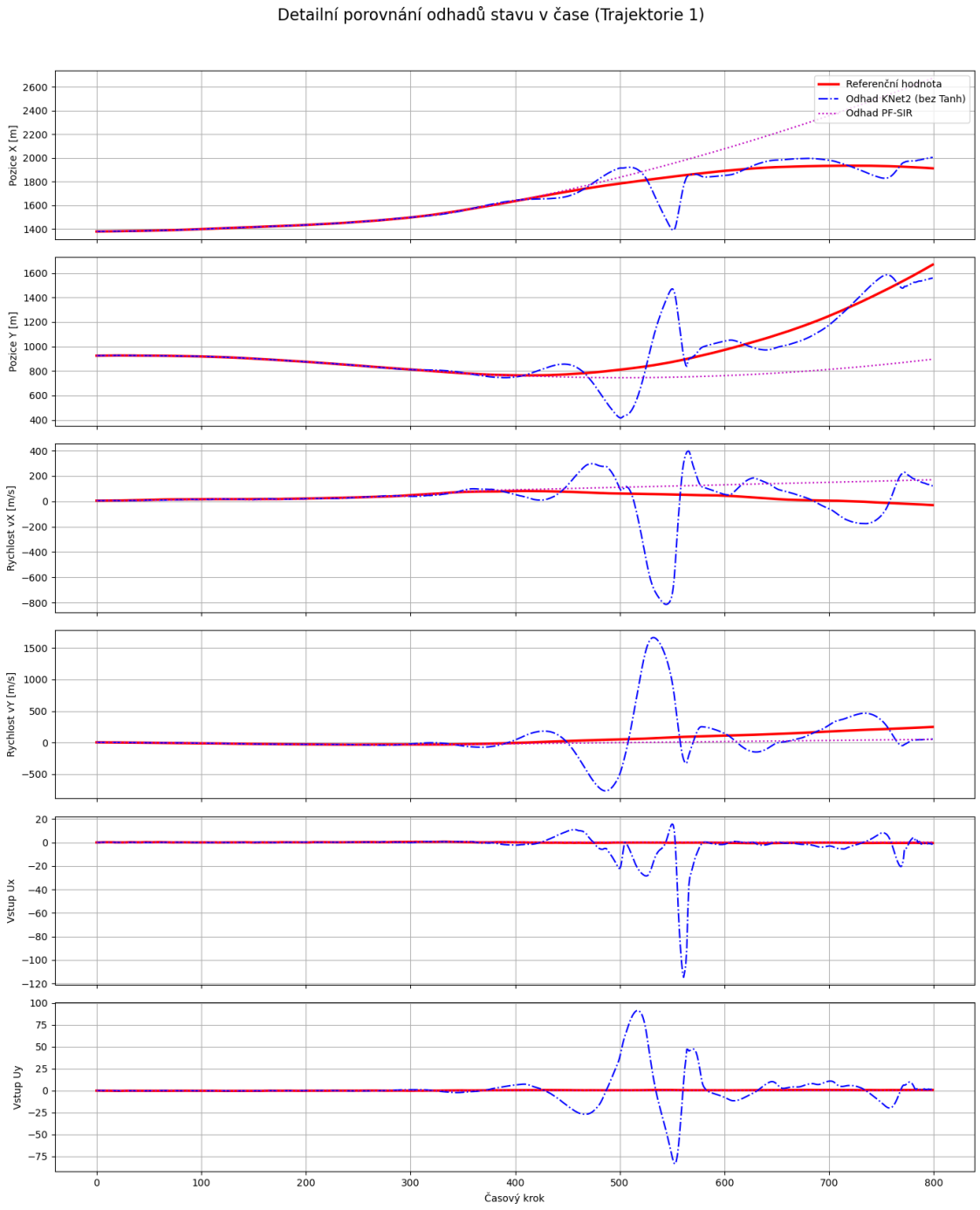
```
# try:
#     knet3_diag_data, knet3_plot_flags = extract_diagnostics(all_knet3_diagnostics_cpu[index], time_axis, gain_time_axis)
#     plot_diag_knet3 = True
# except (NameError, IndexError):
#     print("Varování: Diagnostika pro 'KNet3' nenalezena. Grafy nebudou vykresleny.")
#     plot_diag_knet3 = False


# --- Popisky grafů (Labels) ---
state_dim = x_true_plot.shape[1] # Mělo by být 6
obs_dim = all_x_hat_pf_sir_ideal_cpu[index].shape[1] if len(all_x_hat_pf_sir_ideal_cpu) > 0 else 2 # Placeholder

state_labels = [
    'Pozice X [m]',
    'Pozice Y [m]',
    'Rychlost vX [m/s]',
    'Rychlost vY [m/s]',
    'Vstup Ux',
    'Vstup Uy'
]
error_labels = [f'RMSE {label}' for label in state_labels]

gain_labels_col0 = [f'K[{i},0] (Azimut -> {state_labels[i].split(" ")[0]})' for i in range(state_dim)]
gain_labels_col1 = [f'K[{i},1] (Vzdál. -> {state_labels[i].split(" ")[0]})' for i in range(state_dim)]

diagnostic_labels = {
    'h_norm': 'L2 Norma skrytých stavů $h_t$ (Q+Sigma+S)',
    'innov_norm': 'L2 Norma inovace $\Delta y_t$',
    'inputs_norm': 'L2 Norma vstupních rysů $F_{1..4}$'
}


# ==============================================================================
# 2. VYTVOŘENÍ GRAFŮ
# ==============================================================================
print("Vykresluji grafy...")

# --- Graf 1: Trajektorie (Porovnání všech) ---
fig1, axes1 = plt.subplots(state_dim, 1, figsize=(14, 18), sharex=True)
```

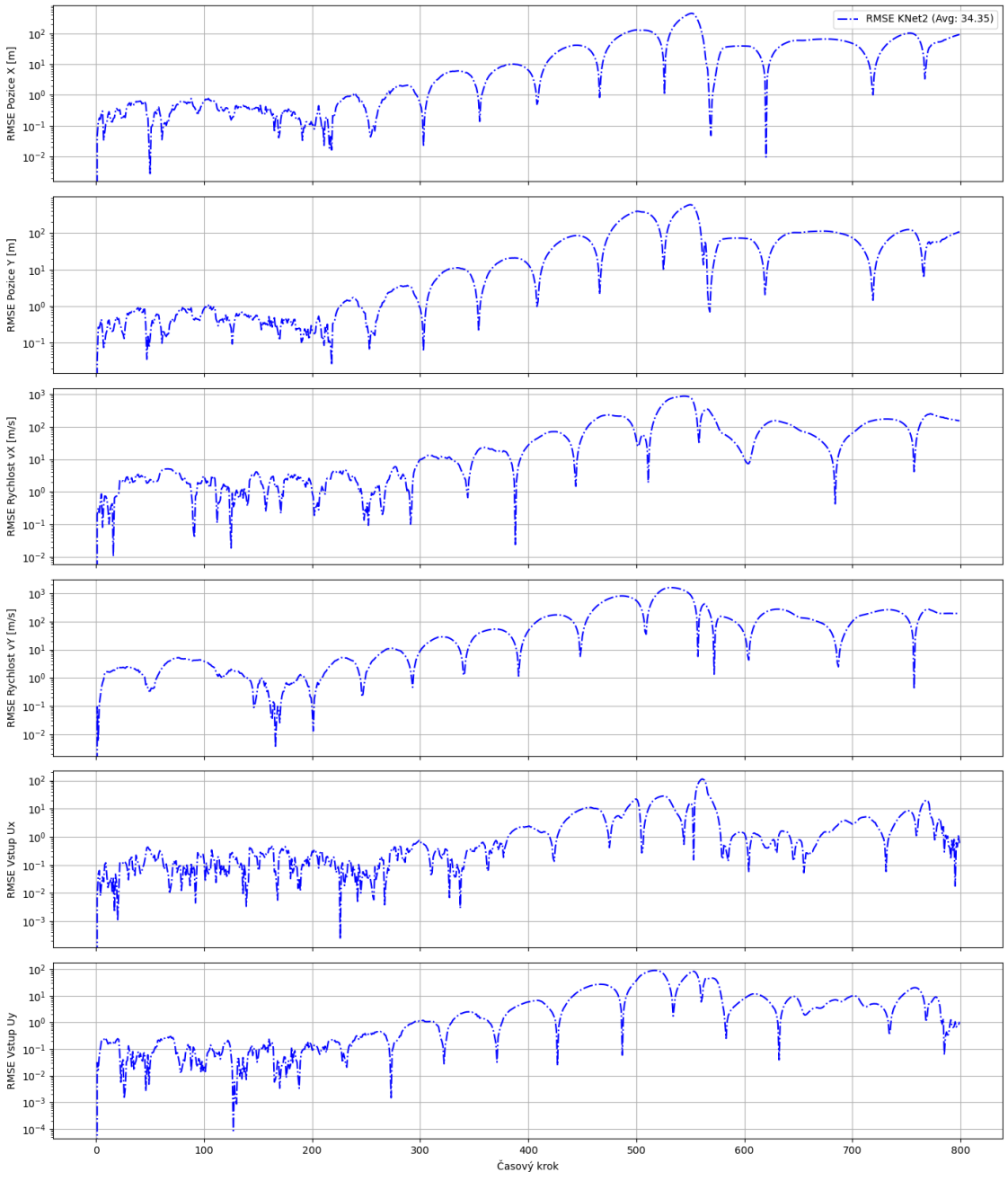
```
fig1.suptitle(f'Detailní porovnání odhadů stavu v čase (Trajektorie␣
 ↪{index+1})', fontsize=16)

# --- Graf 2: RMSE (Porovnání KNet2 vs KNet3) ---
fig2, axes2 = plt.subplots(state_dim, 1, figsize=(14, 18), sharex=True)
fig2.suptitle(f'Porovnání RMSE v čase (Trajektorie {index+1})', fontsize=16)

# --- Smyčka přes všech 6 složek stavu ---
for i in range(state_dim):
    # --- Graf 1: Trajektorie ---
    ax1 = axes1[i]
    ax1.plot(time_axis, x_true_plot[:, i], 'r-', linewidth=2.5,␣
 ↪label='Referenční hodnota')
    ax1.plot(time_axis, x_knet2_tensor[:, i].numpy(), 'b-.', linewidth=1.5,␣
 ↪label='Odhad KNet2 (bez Tanh)')
    # ax1.plot(time_axis, x_knet3_tensor[:, i].numpy(), 'g:', linewidth=2.0,␣
 ↪label='Odhad KNet3 (s Tanh)')

    # PF-SIR vykreslíme, jen pokud má stejný rozměr stavu
    if x_pf_tensor.shape[1] == state_dim:
        ax1.plot(time_axis, x_pf_tensor[:, i].numpy(), 'm:', linewidth=1.5,␣
 ↪label='Odhad PF-SIR')

    ax1.set_ylabel(state_labels[i])
    ax1.grid(True)

    # --- Graf 2: Chyba (RMSE) ---
    ax2 = axes2[i]
    ax2.plot(time_axis, rmse_knet2_per_step[:, i], 'b-.', linewidth=1.5,␣
 ↪label=f'RMSE KNet2 (Avg: {np.mean(rmse_knet2_per_step[1:, i]):.2f})')
    # ax2.plot(time_axis, rmse_knet3_per_step[:, i], 'g:', linewidth=2.0,␣
 ↪label=f'RMSE KNet3 (Avg: {np.mean(rmse_knet3_per_step[1:, i]):.2f})')
    ax2.set_ylabel(error_labels[i])
    ax2.grid(True)
    ax2.set_yscale('log') # Logaritmická osa pro lepší viditelnost velkých chyb

axes1[0].legend(loc='upper right')
axes1[-1].set_xlabel('Časový krok')
axes2[0].legend(loc='upper right')
axes2[-1].set_xlabel('Časový krok')

fig1.tight_layout(rect=[0, 0.03, 1, 0.96])
fig2.tight_layout(rect=[0, 0.03, 1, 0.96])


# --- Funkce pro vykreslení plné diagnostiky ---
```

```
def plot_full_diagnostics(model_name, diag_data, plot_flags, time_axis,␣
 ↪gain_time_axis):
    if not any(plot_flags.values()):
        print(f"Nebylo co vykreslit pro {model_name}.")
        return

    # Vytvoříme 4 hlavní subgrafy
    fig, axes = plt.subplots(4, 1, figsize=(14, 20), sharex=True)
    fig.suptitle(f'Hloubková diagnostika modelu: {model_name} (Trajektorie␣
 ↪{index+1})', fontsize=16)

    (ax_gain0, ax_gain1, ax_h, ax_innov) = axes

    # --- Graf 1: Zisky K (reakce na Azimut) ---
    if plot_flags['gains']:
        gains_to_plot = diag_data['gains_col0_np']
        for i in range(state_dim):
            ax_gain0.plot(gain_time_axis, gains_to_plot[:, i], linewidth=1.5,␣
 ↪label=f'{gain_labels_col0[i]} (Avg: {np.mean(gains_to_plot[:, i]):.4f})')
        ax_gain0.set_ylabel('Kalmanův zisk K[:, 0] (z Azimutu)')
        ax_gain0.grid(True)
        ax_gain0.legend(loc='center left', bbox_to_anchor=(1, 0.5))

    # --- Graf 2: Zisky K (reakce na Vzdálenost) ---
    if plot_flags['gains']:
        gains_to_plot = diag_data['gains_col1_np']
        for i in range(state_dim):
            ax_gain1.plot(gain_time_axis, gains_to_plot[:, i], linewidth=1.5,␣
 ↪label=f'{gain_labels_col1[i]} (Avg: {np.mean(gains_to_plot[:, i]):.4f})')
        ax_gain1.set_ylabel('Kalmanův zisk K[:, 1] (ze Vzdálenosti)')
        ax_gain1.grid(True)
        ax_gain1.legend(loc='center left', bbox_to_anchor=(1, 0.5))

    # --- Graf 3: Norma skrytého stavu h_t ---
    if plot_flags['h_norm']:
        ax_h.plot(time_axis, diag_data['h_norms_np'], 'darkorange', linewidth=1.
 ↪5, label=f'Norma $h_t$')
        ax_h.set_ylabel(diagnostic_labels['h_norm'])
        ax_h.grid(True)
        ax_h.legend(loc='upper right')
        ax_h.set_yscale('log') # Skrytý stav často exploduje

    # --- Graf 4: Norma Inovace vs. Vstupních rysů ---
    if plot_flags['innov_norm']:
        ax_innov.plot(gain_time_axis, diag_data['innov_norms_np'], 'purple',␣
 ↪linewidth=2.0, label=f'Norma Inovace (Avg: {np.
 ↪mean(diag_data["innov_norms_np"]):.2f})')
```

```
        ax_innov.set_ylabel(diagnostic_labels['innov_norm'], color='purple')
        ax_innov.tick_params(axis='y', labelcolor='purple')
        ax_innov.grid(True)
        ax_innov.set_yscale('log')
        ax_innov.legend(loc='upper left')

    if plot_flags['inputs_norm']:
        # Vytvoříme druhou osu Y
        ax_inputs = ax_innov.twinx()
        ax_inputs.plot(gain_time_axis, diag_data['inputs_norms_np'], 'teal',␣
 ↪linestyle=':', linewidth=2.0, label=f'Norma Vstupních Rysů (Avg: {np.
 ↪mean(diag_data["inputs_norms_np"]):.2f})')
        ax_inputs.set_ylabel(diagnostic_labels['inputs_norm'], color='teal')
        ax_inputs.tick_params(axis='y', labelcolor='teal')
        ax_inputs.set_yscale('log')
        ax_inputs.legend(loc='upper right')

    ax_innov.set_xlabel('Časový krok')
    fig.tight_layout(rect=[0, 0.03, 0.85, 0.96]) # Uděláme místo pro legendy␣
 ↪zisků

# --- Vykreslení diagnostických grafů ---
if plot_diag_knet2:
    plot_full_diagnostics("KNet2 (bez Tanh)", knet2_diag_data,␣
 ↪knet2_plot_flags, time_axis, gain_time_axis)

# if plot_diag_knet3:
#     plot_full_diagnostics("KNet3 (s Tanh)", knet3_diag_data,␣
 ↪knet3_plot_flags, time_axis, gain_time_axis)

plt.show()
```

```
Připravuji data pro vykreslení…
Varování: Diagnostika pro 'KNet2' nenalezena. Grafy nebudou vykresleny.
Vykresluji grafy…
```


Detailní porovnání odhadů stavu v čase (Trajektorie 1)
Referenční hodnota
Odhad KNet2 (bez Tanh)
Odhad PF-SIR
Pozice X [m]
Pozice Y [m]
Rychlost vX [m/s]
Rychlost vY [m/s]
Vstup Ux
Vstup Uy
Časový krok



[ ]:
```
# import matplotlib.pyplot as plt
# import numpy as np

# %matplotlib widget

# # --- Předpokládáme, že tyto proměnné již existují z vašeho vyhodnocení ---
```

```
# # all_x_true_cpu: Seznam s pravdivou trajektorií
# # full_x_hat_classic_knet: Tenzor s odhady z klasického KNetu
# # full_x_hat_bkn: Tenzor s odhady z Bayesian KNetu (předpoklad)
# # souradniceX_mapa, souradniceY_mapa, souradniceZ_mapa: Data mapy
# # terMap: Vaše interpolační funkce

# # --- Krok 1: Příprava dat ---
# x_true_plot = all_x_true_cpu[0].numpy()
# x_knet_plot = full_x_hat_classic_knet.cpu().numpy()

# print(f"Tvar skutečné trajektorie: {x_true_plot.shape}")
# print(f"Tvar odhadnuté trajektorie (KNet): {x_knet_plot.shape}")

# # --- Krok 2: Vytvoření 3D grafu ---
# fig = plt.figure(figsize=(14, 12))
# ax = fig.add_subplot(111, projection='3d')

# # Vykreslení povrchu terénu (volitelné)
# ax.plot_surface(souradniceX_mapa, souradniceY_mapa, souradniceZ_mapa,
#                 rstride=100, cstride=100, cmap='terrain', alpha=0.3)

# # --- Krok 3: Vykreslení trajektorií ---

# # A. Skutečná (referenční) trajektorie
# px_true = x_true_plot[:, 0]
# py_true = x_true_plot[:, 1]
# pz_true = terMap(px_true, py_true)
# ax.plot(px_true, py_true, pz_true, 'r-', linewidth=3, label='Referenční␣
 ↪trajektorie')

# # B. Odhadnutá trajektorie z KalmanNetu
# px_knet = x_knet_plot[:, 0]
# py_knet = x_knet_plot[:, 1]
# pz_knet = terMap(px_knet, py_knet)
# ax.plot(px_knet, py_knet, pz_knet, 'g--', linewidth=3, label='Odhad KNet')


# # --- Krok 4: Finalizace grafu ---
# ax.plot([px_true[0]], [py_true[0]], [pz_true[0]],
#         'o', color='black', markersize=10, label='Start')

# ax.set_xlabel('Souřadnice X [m]')
# ax.set_ylabel('Souřadnice Y [m]')
# ax.set_zlabel('Nadmořská výška Z [m]')

# # Upravíme název, aby zahrnoval všechny modely
# ax.set_title('Porovnání referenční trajektorie a odhadů KNet/BKN')
```

```
# ax.legend()
# ax.grid(True)

# ax.view_init(elev=30., azim=-60)

# plt.show()
```

[ ]:
```
# import matplotlib.pyplot as plt
# import numpy as np

# %matplotlib widget

# # --- Předpokládáme, že tyto proměnné již existují z vašeho vyhodnocení ---
# # all_x_true_cpu: Seznam s pravdivou trajektorií
# # full_x_hat_classic_knet: Tenzor s odhady z klasického KNetu
# # full_x_hat_bkn: Tenzor s odhady z Bayesian KNetu (předpoklad)
# # souradniceX_mapa, souradniceY_mapa, souradniceZ_mapa: Data mapy
# # terMap: Vaše interpolační funkce

# # --- Krok 1: Příprava dat ---
# x_true_plot = all_x_true_cpu[0].numpy()

# print(f"Tvar skutečné trajektorie: {x_true_plot.shape}")


# # --- Krok 2: Vytvoření 3D grafu ---
# fig = plt.figure(figsize=(14, 12))
# ax = fig.add_subplot(111, projection='3d')

# # Vykreslení povrchu terénu (volitelné)
# ax.plot_surface(souradniceX_mapa, souradniceY_mapa, souradniceZ_mapa,
#                 rstride=100, cstride=100, cmap='terrain', alpha=0.3)

# # --- Krok 3: Vykreslení trajektorií ---

# # A. Skutečná (referenční) trajektorie
# px_true = x_true_plot[:, 0]
# py_true = x_true_plot[:, 1]
# pz_true = terMap(px_true, py_true)
# ax.plot(px_true, py_true, pz_true, 'r-', linewidth=3, label='Referenční␣
 ↪trajektorie')


# # --- Krok 4: Finalizace grafu ---
# ax.plot([px_true[0]], [py_true[0]], [pz_true[0]],
#         'o', color='black', markersize=10, label='Start')
```

```
# ax.set_xlabel('Souřadnice X [m]')
# ax.set_ylabel('Souřadnice Y [m]')
# ax.set_zlabel('Nadmořská výška Z [m]')

# # Upravíme název, aby zahrnoval všechny modely
# ax.set_title('Porovnání referenční trajektorie a odhadů KNet/BKN')
# ax.legend()
# ax.grid(True)

# ax.view_init(elev=30., azim=-60)

# plt.show()
```