

4D_system

October 30, 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 reprodukovanost (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

```

[3]: mat_data = loadmat(dataset_path)

souradniceX_mapa = mat_data['souradniceX']
souradniceY_mapa = mat_data['souradniceY']
souradniceZ_mapa = mat_data['souradniceZ']
souradniceGNSS = mat_data['souradniceGNSS']

# --- KROK 1: Extrakce 1D os z 2D mřížek ---
# Získáme unikátní souřadnice pro osy X a Y.
# Pro osu X vezmeme první řádek z X matice.
# Pro osu Y vezmeme první sloupec z Y matice.
x_axis_unique = souradniceX_mapa[0, :]
y_axis_unique = souradniceY_mapa[:, 0]

print(f"Rozměry 1D osy X: {x_axis_unique.shape}")
print(f"Rozměry 1D osy Y: {y_axis_unique.shape}")
print(f"Rozměry 2D dat výšek Z: {souradniceZ_mapa.shape}")

# --- KROK 2: Vytvoření interpolačního objektu ---
# POZOR: Scipy očekává, že osy budou v pořadí (y, x), protože
# NumPy pole jsou indexována jako (řádek, sloupec), což odpovídá (y, x).
print("\nVytvářím interpolační funkci...")
terMap_interpolator = RegularGridInterpolator(
    (y_axis_unique, x_axis_unique), # N-tice 1D os (nejprve Y, pak X)
    souradniceZ_mapa,
    bounds_error=False, # NEVYHAZUJ CHYBU
    fill_value=np.nan
)

```

```

print("...interpoláční funkce vytvořena.")

# --- KROK 3: Vytvoření finální, uživatelsky přívětivé funkce ---
def terMap(px, py):
    """
    Vypočítá nadmořskou výšku pro dané souřadnice (px, py)
    pomocí interpolace z mapy terénu.

    Funkce zvládne jak jednotlivé body, tak celé pole bodů.
    """
    # Spojíme vstupní body do formátu, kterému interpolátor rozumí:
    # pole o dvou sloupcích [y, x].
    points_to_query = np.column_stack((py, px))

    # Zavoláme interpolátor a vrátíme výsledek
    return terMap_interpolator(points_to_query)

```

Rozměry 1D osy X: (2500,)
Rozměry 1D osy Y: (2500,)
Rozměry 2D dat výšek Z: (2500, 2500)

Vytvářím interpoláční funkci...
...interpoláční funkce vytvořena.

1 4D model

```
[4]: import torch
from math import pi
from Systems import DynamicSystemTAN

state_dim = 4
obs_dim = 3
dT = 1
q = 1

F = torch.tensor([[1.0, 0.0, dT, 0.0],
                  [0.0, 1.0, 0.0, dT],
                  [0.0, 0.0, 1.0, 0.0],
                  [0.0, 0.0, 0.0, 1.0]])

Q = q* torch.tensor([[dT**3/3, 0.0, dT**2/2, 0.0],
                     [0.0, dT**3/3, 0.0, dT**2/2],
                     [dT**2/2, 0.0, dT, 0.0],
                     [0.0, dT**2/2, 0.0, dT]])

R = torch.tensor([[3.0**2, 0.0, 0.0],
                  [0.0, 1.0**2, 0.0],
                  [0.0, 0.0, 1.0**2]])
```

```

initial_velocity_np = souradniceGNSS[:, 1] - souradniceGNSS[:, 0]
initial_velocity = torch.from_numpy(initial_velocity_np)

initial_position = torch.from_numpy(souradniceGNSS[:, 0])
x_0 = torch.cat([
    initial_position,
    initial_velocity
]).float()
print(x_0)

P_0 = torch.tensor([[25.0, 0.0, 0.0, 0.0],
                   [0.0, 25.0, 0.0, 0.0],
                   [0.0, 0.0, 0.5, 0.0],
                   [0.0, 0.0, 0.0, 0.5]])

def h_nl_robust(x: torch.Tensor) -> torch.Tensor:
    # ... (implementace s clampingem, jak jsme si ukázali dříve) ...
    # Získání hranic mapy
    min_x, max_x = x_axis_unique.min(), x_axis_unique.max()
    min_y, max_y = y_axis_unique.min(), y_axis_unique.max()

    # Oříznutí pozic POUZE pro dotaz do mapy
    px_safe = x[:, 0].clone().clamp(min_x, max_x)
    py_safe = x[:, 1].clone().clamp(min_y, max_y)
    vyska_terenu_np = terMap(px_safe.detach().cpu().numpy(), py_safe.detach().
    ↵cpu().numpy())
    vyska_terenu = torch.from_numpy(vyska_terenu_np).float().to(x.device)

    # Zbytek výpočtu s původními rychlostmi
    eps = 1e-12
    vx_w, vy_w = x[:, 2], x[:, 3]
    norm_v_w = torch.sqrt(vx_w**2 + vy_w**2).clamp(min=eps)
    cos_psi = vx_w / norm_v_w
    sin_psi = vy_w / norm_v_w
    vx_b = cos_psi * vx_w - sin_psi * vy_w
    vy_b = sin_psi * vx_w + cos_psi * vy_w

    result = torch.stack([vyska_terenu, vx_b, vy_b], dim=1)

    # Pojistka pro případ, že by terMap přesto vrátila NaN
    if torch.isnan(result).any():
        print("Varování: NaN hodnoty v měření detekovány, nahrazuji nulami.")
        result[torch.isnan(result)] = 0

    return result

```

```

x_axis_unique = souradniceX_mapa[0, :]
y_axis_unique = souradniceY_mapa[:, 0]

print("Vytvářím instanci DynamicSystemTAN...")
system_model = DynamicSystemTAN(
    state_dim=state_dim,
    obs_dim=obs_dim,
    Q=Q.float(),
    R=R.float(),
    Ex0=x_0.float(),
    P0=P_0.float(),
    F=F.float(),
    h=h_nl_robust,
    x_axis_unique=x_axis_unique,
    y_axis_unique=y_axis_unique,
    device=device
)

```

tensor([1.4875e+06, 6.3955e+06, 4.3225e+00, -4.1456e+01])

Vytvářím instanci DynamicSystemTAN...

INFO: DynamicSystemTAN inicializován s hranicemi mapy:

X: [1476611.42, 1489541.47]

Y: [6384032.63, 6400441.34]

```

[5]: # TRAIN_SEQ_LEN = 300      # Krátké sekvence pro stabilní trénink (TBPTT)
# VALID_SEQ_LEN = 400        # Stejná délka pro konzistentní validaci
# # TEST_SEQ_LEN = 200       # Dlouhé sekvence pro testování generalizace

# NUM_TRAIN_TRAJ = 100      # Hodně trénovacích příkladů
# NUM_VALID_TRAJ = 10        # Dostatek pro spolehlivou validaci
# # NUM_TEST_TRAJ = 100      # Pro robustní vyhodnocení

# BATCH_SIZE = 16            # Dobrý kompromis

# x_train, y_train = utils.generate_data_for_map(system_model, □
#     ↪num_trajectories=NUM_TRAIN_TRAJ, seq_len=TRAIN_SEQ_LEN)
# x_val, y_val = utils.generate_data_for_map(system_model, □
#     ↪num_trajectories=NUM_VALID_TRAJ, seq_len=VALID_SEQ_LEN)
# # x_test, y_test = utils.generate_data_for_map(system_model, □
#     ↪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)

```

```
[6]: import torch
from torch.utils.data import TensorDataset, DataLoader
from Systems import DynamicSystemTAN # Nebo jak se jmenuje tvůj model
from copy import deepcopy # Pro vytvoření kopie modelu
import numpy as np
import random # Pro náhodný výběr indexu

# --- Konfigurace (stejná jako předtím) ---
TRAIN_SEQ_LEN = 200
VALID_SEQ_LEN = 300
NUM_TRAIN_SETS = 20 # Kolik různých náhodných počátečních podmínek pro trénink
NUM_VALID_SETS = 20 # Kolik různých náhodných počátečních podmínek pro validaci
TRAJ_PER_SET_TRAIN = 12 # Kolik trajektorií na jednu počáteční podmínsku
TRAJ_PER_SET_VALID = 1 # Kolik trajektorií na jednu počáteční podmínsku
BATCH_SIZE = 8
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

# --- Načti souradniceGNSS (předpokládá se, že už existuje) ---
# souradniceGNSS = mat_data['souradniceGNSS'] # Pole tvaru [rozměr, čas]
num_gnss_points = souradniceGNSS.shape[1]
print(f"Načteno {num_gnss_points} GNSS bodů pro výběr počátečních podmínek.")

# --- Zde definuj tvůj původní system_model ---
# Předpokládám, že existuje proměnná 'system_model' s nějakými výchozími Ex0 a_0
original_system_model = system_model # Uchovej si originál

# --- Definuj varianty pro PO (volitelné, můžeš použít jen jednu) ---
default_PO = torch.diag(torch.tensor([25.0, 25.0, 0.5, 0.5], device=device)).float()
# Můžeš přidat další PO varianty sem, pokud chceš větší variabilitu
# PO_options = [
#     torch.diag(torch.tensor([50.0, 50.0, 1.0, 1.0], device=device)).float(),
#     torch.diag(torch.tensor([100.0, 100.0, 2.0, 2.0], device=device)).float(),
#     torch.diag(torch.tensor([25.0, 25.0, 0.5, 0.5], device=device)).float()
# ]

# --- Generování trénovacích dat ---
print("Generuji trénovací data s náhodnými počátečními podmínkami z GNSS...")
all_x_train = []
all_y_train = []
for i in range(NUM_TRAIN_SETS):
    print(f"  Generuji sadu {i+1}/{NUM_TRAIN_SETS}...")

    # 1. Náhodně vyber startovní index t (musí být t+1 platný index)
```

```

start_index = random.randint(0, num_gnss_points - 2)

# 2. Vypočítej Ex0 z GNSS dat
initial_pos_np = souradniceGNSS[:2, start_index]
next_pos_np = souradniceGNSS[:2, start_index + 1]
# Výpočet rychlosti jako rozdíl pozic (předpokládá dT=1 mezi GNSS body, u
↪uprav pokud ne)
# Pokud dT pro GNSS není 1, děl rychlosť tímto dT
initial_vel_np = next_pos_np - initial_pos_np
Ex0_sampled = torch.cat([
    torch.from_numpy(initial_pos_np),
    torch.from_numpy(initial_vel_np)
]).float().to(device)

# 3. Vyber P0 (zde používáme defaultní, můžeš přidat random.
↪choice(P0_options))
P0_current = default_P0

# 4. Vytvoř dočasný model a generuj data
temp_model = deepcopy(original_system_model)
temp_model.Ex0 = Ex0_sampled
temp_model.P0 = P0_current

print(f"    Startovní index: {start_index}, Ex0: {Ex0_sampled.cpu() .
↪numpy()}") # Výpis pro kontrolu

x_batch, y_batch = utils.generate_data_for_map(
    temp_model,
    num_trajectories=TRAJ_PER_SET_TRAIN,
    seq_len=TRAIN_SEQ_LEN
)
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 (analogicky) ---
print("Generuje validační data s náhodnými počátečními podmínkami z GNSS...")
all_x_val = []
all_y_val = []
for i in range(NUM_VALID_SETS):
    print(f"    Generuje sadu {i+1}/{NUM_VALID_SETS}...")
    start_index = random.randint(0, num_gnss_points - 2)
    initial_pos_np = souradniceGNSS[:2, start_index]
    next_pos_np = souradniceGNSS[:2, start_index + 1]

```

```

    initial_vel_np = next_pos_np - initial_pos_np # Uprav dělení  $dT$ , pokud je u
    ↪potřeba
    Ex0_sampled = torch.cat([
        torch.from_numpy(initial_pos_np),
        torch.from_numpy(initial_vel_np)
    ]).float().to(device)
    P0_current = default_P0 # Nebo random.choice(P0_options)

    temp_model = deepcopy(original_system_model)
    temp_model.Ex0 = Ex0_sampled
    temp_model.P0 = P0_current

    print(f"      Startovní index: {start_index}, Ex0: {Ex0_sampled.cpu() .
    ↪numpy()}")
}

x_batch, y_batch = utils.generate_data_for_map(
    temp_model,
    num_trajectories=TRAJ_PER_SET_VALID,
    seq_len=VALID_SEQ_LEN
)
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}")

# --- 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.")

```

Načteno 1276 GNSS bodů pro výběr počátečních podmínek.
 Generuje trénovací data s náhodnými počátečními podmínkami z GNSS...
 Generuje sadu 1/20...

Startovní index: 228, Ex0: [1.4852552e+06 6.3877955e+06 -9.9809055e+00
 -2.1696117e+01]
 INFO: Generátor dat používá hranice X:[1476611.42-1489541.47],
 Y:[6384032.63-6400441.34]
 Generuje 12 platných trajektorií (metoda zahodení)...
 Úspěšně vygenerována trajektorie 1/12 (Pokusů: 3)
 Úspěšně vygenerována trajektorie 2/12 (Pokusů: 5)
 Úspěšně vygenerována trajektorie 3/12 (Pokusů: 6)

Úspěšně vygenerována trajektorie 4/12 (Pokusů: 9)
Úspěšně vygenerována trajektorie 5/12 (Pokusů: 12)
Úspěšně vygenerována trajektorie 6/12 (Pokusů: 17)
Úspěšně vygenerována trajektorie 7/12 (Pokusů: 18)
Úspěšně vygenerována trajektorie 8/12 (Pokusů: 23)
Úspěšně vygenerována trajektorie 9/12 (Pokusů: 24)
Úspěšně vygenerována trajektorie 10/12 (Pokusů: 27)
Úspěšně vygenerována trajektorie 11/12 (Pokusů: 28)
Úspěšně vygenerována trajektorie 12/12 (Pokusů: 32)

Generování dat dokončeno.

Celkový počet pokusů: 32

Úspěšnost (platné trajektorie / pokusy): 37.50%

Celkový počet vygenerovaných trajektorií: torch.Size([12, 200, 4])

Generuje sadu 2/20...

Startovní index: 51, Ex0: [1.4879221e+06 6.3935160e+06 7.8290997e+00
-3.7429684e+01]

INFO: Generátor dat používá hranice X:[1476611.42-1489541.47],
Y: [6384032.63-6400441.34]

Generuje 12 platných trajektorií (metoda zahodení)...

Úspěšně vygenerována trajektorie 1/12 (Pokusů: 2)
Úspěšně vygenerována trajektorie 2/12 (Pokusů: 8)
Úspěšně vygenerována trajektorie 3/12 (Pokusů: 9)
Úspěšně vygenerována trajektorie 4/12 (Pokusů: 14)
Úspěšně vygenerována trajektorie 5/12 (Pokusů: 21)
Úspěšně vygenerována trajektorie 6/12 (Pokusů: 26)
Úspěšně vygenerována trajektorie 7/12 (Pokusů: 27)
Úspěšně vygenerována trajektorie 8/12 (Pokusů: 28)
Úspěšně vygenerována trajektorie 9/12 (Pokusů: 30)
Úspěšně vygenerována trajektorie 10/12 (Pokusů: 31)
Úspěšně vygenerována trajektorie 11/12 (Pokusů: 33)
Úspěšně vygenerována trajektorie 12/12 (Pokusů: 39)

Generování dat dokončeno.

Celkový počet pokusů: 39

Úspěšnost (platné trajektorie / pokusy): 30.77%

Celkový počet vygenerovaných trajektorií: torch.Size([12, 200, 4])

Generuje sadu 3/20...

Startovní index: 563, Ex0: [1.4795671e+06 6.3884300e+06 2.2263899e+01
1.5478781e+01]

INFO: Generátor dat používá hranice X:[1476611.42-1489541.47],
Y: [6384032.63-6400441.34]

Generuje 12 platných trajektorií (metoda zahodení)...

Úspěšně vygenerována trajektorie 1/12 (Pokusů: 1)
Úspěšně vygenerována trajektorie 2/12 (Pokusů: 2)
Úspěšně vygenerována trajektorie 3/12 (Pokusů: 3)
Úspěšně vygenerována trajektorie 4/12 (Pokusů: 4)
Úspěšně vygenerována trajektorie 5/12 (Pokusů: 5)

Úspěšně vygenerována trajektorie 6/12 (Pokusů: 6)
Úspěšně vygenerována trajektorie 7/12 (Pokusů: 7)
Úspěšně vygenerována trajektorie 8/12 (Pokusů: 8)
Úspěšně vygenerována trajektorie 9/12 (Pokusů: 9)
Úspěšně vygenerována trajektorie 10/12 (Pokusů: 10)
Úspěšně vygenerována trajektorie 11/12 (Pokusů: 11)
Úspěšně vygenerována trajektorie 12/12 (Pokusů: 12)

Generování dat dokončeno.

Celkový počet pokusů: 12

Úspěšnost (platné trajektorie / pokusy): 100.00%

Celkový počet vygenerovaných trajektorií: torch.Size([12, 200, 4])

Generuje sadu 4/20...

Startovní index: 501, Ex0: [1.4785838e+06 6.3871215e+06 4.6019478e+00
1.8800343e+01]

INFO: Generátor dat používá hranice X:[1476611.42-1489541.47],
Y:[6384032.63-6400441.34]

Generuje 12 platných trajektorií (metoda zahodení)...

Úspěšně vygenerována trajektorie 1/12 (Pokusů: 1)
Úspěšně vygenerována trajektorie 2/12 (Pokusů: 2)
Úspěšně vygenerována trajektorie 3/12 (Pokusů: 3)
Úspěšně vygenerována trajektorie 4/12 (Pokusů: 4)
Úspěšně vygenerována trajektorie 5/12 (Pokusů: 5)
Úspěšně vygenerována trajektorie 6/12 (Pokusů: 6)
Úspěšně vygenerována trajektorie 7/12 (Pokusů: 8)
Úspěšně vygenerována trajektorie 8/12 (Pokusů: 9)
Úspěšně vygenerována trajektorie 9/12 (Pokusů: 10)
Úspěšně vygenerována trajektorie 10/12 (Pokusů: 11)
Úspěšně vygenerována trajektorie 11/12 (Pokusů: 12)
Úspěšně vygenerována trajektorie 12/12 (Pokusů: 13)

Generování dat dokončeno.

Celkový počet pokusů: 13

Úspěšnost (platné trajektorie / pokusy): 92.31%

Celkový počet vygenerovaných trajektorií: torch.Size([12, 200, 4])

Generuje sadu 5/20...

Startovní index: 457, Ex0: [1.4792504e+06 6.3867840e+06 -1.9517647e+01
1.3154374e+01]

INFO: Generátor dat používá hranice X:[1476611.42-1489541.47],
Y:[6384032.63-6400441.34]

Generuje 12 platných trajektorií (metoda zahodení)...

Úspěšně vygenerována trajektorie 1/12 (Pokusů: 6)
Úspěšně vygenerována trajektorie 2/12 (Pokusů: 13)
Úspěšně vygenerována trajektorie 3/12 (Pokusů: 29)
Úspěšně vygenerována trajektorie 4/12 (Pokusů: 44)
Úspěšně vygenerována trajektorie 5/12 (Pokusů: 49)
Úspěšně vygenerována trajektorie 6/12 (Pokusů: 50)
Úspěšně vygenerována trajektorie 7/12 (Pokusů: 55)

Úspěšně vygenerována trajektorie 8/12 (Pokusů: 62)
Úspěšně vygenerována trajektorie 9/12 (Pokusů: 75)
Úspěšně vygenerována trajektorie 10/12 (Pokusů: 76)
Úspěšně vygenerována trajektorie 11/12 (Pokusů: 84)
Úspěšně vygenerována trajektorie 12/12 (Pokusů: 85)

Generování dat dokončeno.

Celkový počet pokusů: 85

Úspěšnost (platné trajektorie / pokusy): 14.12%

Celkový počet vygenerovaných trajektorií: torch.Size([12, 200, 4])

Generuje sadu 6/20...

Startovní index: 285, Ex0: [1.4837711e+06 6.3872025e+06 -3.5696819e+01
-1.9632772e+01]

INFO: Generátor dat používá hranice X:[1476611.42-1489541.47],
Y:[6384032.63-6400441.34]

Generuje 12 platných trajektorií (metoda zahodení)...

Úspěšně vygenerována trajektorie 1/12 (Pokusů: 2)

Úspěšně vygenerována trajektorie 2/12 (Pokusů: 3)

Úspěšně vygenerována trajektorie 3/12 (Pokusů: 4)

Úspěšně vygenerována trajektorie 4/12 (Pokusů: 9)

Úspěšně vygenerována trajektorie 5/12 (Pokusů: 14)

Úspěšně vygenerována trajektorie 6/12 (Pokusů: 26)

Úspěšně vygenerována trajektorie 7/12 (Pokusů: 32)

Úspěšně vygenerována trajektorie 8/12 (Pokusů: 43)

Úspěšně vygenerována trajektorie 9/12 (Pokusů: 46)

Úspěšně vygenerována trajektorie 10/12 (Pokusů: 64)

Úspěšně vygenerována trajektorie 11/12 (Pokusů: 74)

Úspěšně vygenerována trajektorie 12/12 (Pokusů: 75)

Generování dat dokončeno.

Celkový počet pokusů: 75

Úspěšnost (platné trajektorie / pokusy): 16.00%

Celkový počet vygenerovaných trajektorií: torch.Size([12, 200, 4])

Generuje sadu 7/20...

Startovní index: 209, Ex0: [1.4855200e+06 6.3879445e+06 -1.2059240e+01
-1.6939651e+01]

INFO: Generátor dat používá hranice X:[1476611.42-1489541.47],
Y:[6384032.63-6400441.34]

Generuje 12 platných trajektorií (metoda zahodení)...

Úspěšně vygenerována trajektorie 1/12 (Pokusů: 2)

Úspěšně vygenerována trajektorie 2/12 (Pokusů: 3)

Úspěšně vygenerována trajektorie 3/12 (Pokusů: 6)

Úspěšně vygenerována trajektorie 4/12 (Pokusů: 7)

Úspěšně vygenerována trajektorie 5/12 (Pokusů: 9)

Úspěšně vygenerována trajektorie 6/12 (Pokusů: 10)

Úspěšně vygenerována trajektorie 7/12 (Pokusů: 14)

Úspěšně vygenerována trajektorie 8/12 (Pokusů: 15)

Úspěšně vygenerována trajektorie 9/12 (Pokusů: 18)

Úspěšně vygenerována trajektorie 10/12 (Pokusů: 19)
Úspěšně vygenerována trajektorie 11/12 (Pokusů: 20)
Úspěšně vygenerována trajektorie 12/12 (Pokusů: 22)

Generování dat dokončeno.

Celkový počet pokusů: 22

Úspěšnost (platné trajektorie / pokusy): 54.55%

Celkový počet vygenerovaných trajektorií: torch.Size([12, 200, 4])

Generuji sadu 8/20...

Startovní index: 1116, Ex0: [1.48665962e+06 6.39420500e+06 1.15104353e+00
1.36823635e+01]

INFO: Generátor dat používá hranice X:[1476611.42-1489541.47],
Y:[6384032.63-6400441.34]

Generuji 12 platných trajektorií (metoda zahodení)...

Úspěšně vygenerována trajektorie 1/12 (Pokusů: 1)

Úspěšně vygenerována trajektorie 2/12 (Pokusů: 2)

Úspěšně vygenerována trajektorie 3/12 (Pokusů: 3)

Úspěšně vygenerována trajektorie 4/12 (Pokusů: 4)

Úspěšně vygenerována trajektorie 5/12 (Pokusů: 5)

Úspěšně vygenerována trajektorie 6/12 (Pokusů: 6)

Úspěšně vygenerována trajektorie 7/12 (Pokusů: 7)

Úspěšně vygenerována trajektorie 8/12 (Pokusů: 8)

Úspěšně vygenerována trajektorie 9/12 (Pokusů: 9)

Úspěšně vygenerována trajektorie 10/12 (Pokusů: 10)

Úspěšně vygenerována trajektorie 11/12 (Pokusů: 11)

Úspěšně vygenerována trajektorie 12/12 (Pokusů: 13)

Generování dat dokončeno.

Celkový počet pokusů: 13

Úspěšnost (platné trajektorie / pokusy): 92.31%

Celkový počet vygenerovaných trajektorií: torch.Size([12, 200, 4])

Generuji sadu 9/20...

Startovní index: 178, Ex0: [1.4861084e+06 6.3888310e+06 -2.1021572e+01
-3.2106159e+01]

INFO: Generátor dat používá hranice X:[1476611.42-1489541.47],
Y:[6384032.63-6400441.34]

Generuji 12 platných trajektorií (metoda zahodení)...

Úspěšně vygenerována trajektorie 1/12 (Pokusů: 4)

Úspěšně vygenerována trajektorie 2/12 (Pokusů: 6)

Úspěšně vygenerována trajektorie 3/12 (Pokusů: 8)

Úspěšně vygenerována trajektorie 4/12 (Pokusů: 10)

Úspěšně vygenerována trajektorie 5/12 (Pokusů: 24)

Úspěšně vygenerována trajektorie 6/12 (Pokusů: 26)

Úspěšně vygenerována trajektorie 7/12 (Pokusů: 28)

Úspěšně vygenerována trajektorie 8/12 (Pokusů: 53)

Úspěšně vygenerována trajektorie 9/12 (Pokusů: 54)

Úspěšně vygenerována trajektorie 10/12 (Pokusů: 55)

Úspěšně vygenerována trajektorie 11/12 (Pokusů: 62)

Úspěšně vygenerována trajektorie 12/12 (Pokusů: 63)

Generování dat dokončeno.

Celkový počet pokusů: 63

Úspěšnost (platné trajektorie / pokusy): 19.05%

Celkový počet vygenerovaných trajektorií: torch.Size([12, 200, 4])

Generuje sadu 10/20...

Startovní index: 1209, Ex0: [1.4868909e+06 6.3961370e+06 7.4584059e-02
-2.9264621e-02]

INFO: Generátor dat používá hranice X:[1476611.42-1489541.47],
Y:[6384032.63-6400441.34]

Generuje 12 platných trajektorií (metoda zahodení)...

Úspěšně vygenerována trajektorie 1/12 (Pokusů: 1)

Úspěšně vygenerována trajektorie 2/12 (Pokusů: 2)

Úspěšně vygenerována trajektorie 3/12 (Pokusů: 3)

Úspěšně vygenerována trajektorie 4/12 (Pokusů: 4)

Úspěšně vygenerována trajektorie 5/12 (Pokusů: 5)

Úspěšně vygenerována trajektorie 6/12 (Pokusů: 6)

Úspěšně vygenerována trajektorie 7/12 (Pokusů: 7)

Úspěšně vygenerována trajektorie 8/12 (Pokusů: 8)

Úspěšně vygenerována trajektorie 9/12 (Pokusů: 9)

Úspěšně vygenerována trajektorie 10/12 (Pokusů: 10)

Úspěšně vygenerována trajektorie 11/12 (Pokusů: 11)

Úspěšně vygenerována trajektorie 12/12 (Pokusů: 12)

Generování dat dokončeno.

Celkový počet pokusů: 12

Úspěšnost (platné trajektorie / pokusy): 100.00%

Celkový počet vygenerovaných trajektorií: torch.Size([12, 200, 4])

Generuje sadu 11/20...

Startovní index: 864, Ex0: [1.4835594e+06 6.3946170e+06 1.2616951e+01
3.1811808e+01]

INFO: Generátor dat používá hranice X:[1476611.42-1489541.47],
Y:[6384032.63-6400441.34]

Generuje 12 platných trajektorií (metoda zahodení)...

Úspěšně vygenerována trajektorie 1/12 (Pokusů: 5)

Úspěšně vygenerována trajektorie 2/12 (Pokusů: 8)

Úspěšně vygenerována trajektorie 3/12 (Pokusů: 10)

Úspěšně vygenerována trajektorie 4/12 (Pokusů: 11)

Úspěšně vygenerována trajektorie 5/12 (Pokusů: 15)

Úspěšně vygenerována trajektorie 6/12 (Pokusů: 17)

Úspěšně vygenerována trajektorie 7/12 (Pokusů: 19)

Úspěšně vygenerována trajektorie 8/12 (Pokusů: 20)

Úspěšně vygenerována trajektorie 9/12 (Pokusů: 21)

Úspěšně vygenerována trajektorie 10/12 (Pokusů: 22)

Úspěšně vygenerována trajektorie 11/12 (Pokusů: 23)

Úspěšně vygenerována trajektorie 12/12 (Pokusů: 26)

Generování dat dokončeno.
Celkový počet pokusů: 26
Úspěšnost (platné trajektorie / pokusy): 46.15%
Celkový počet vygenerovaných trajektorií: torch.Size([12, 200, 4])
Generuji sadu 12/20...
Startovní index: 65, Ex0: [1.4880126e+06 6.3929805e+06 3.4130557e+00
-3.9693569e+01]
INFO: Generátor dat používá hranice X:[1476611.42-1489541.47],
Y:[6384032.63-6400441.34]
Generuji 12 platných trajektorií (metoda zahodení)...
Úspěšně vygenerována trajektorie 1/12 (Pokusů: 2)
Úspěšně vygenerována trajektorie 2/12 (Pokusů: 3)
Úspěšně vygenerována trajektorie 3/12 (Pokusů: 6)
Úspěšně vygenerována trajektorie 4/12 (Pokusů: 7)
Úspěšně vygenerována trajektorie 5/12 (Pokusů: 8)
Úspěšně vygenerována trajektorie 6/12 (Pokusů: 9)
Úspěšně vygenerována trajektorie 7/12 (Pokusů: 10)
Úspěšně vygenerována trajektorie 8/12 (Pokusů: 12)
Úspěšně vygenerována trajektorie 9/12 (Pokusů: 13)
Úspěšně vygenerována trajektorie 10/12 (Pokusů: 14)
Úspěšně vygenerována trajektorie 11/12 (Pokusů: 15)
Úspěšně vygenerována trajektorie 12/12 (Pokusů: 16)

Generování dat dokončeno.
Celkový počet pokusů: 16
Úspěšnost (platné trajektorie / pokusy): 75.00%
Celkový počet vygenerovaných trajektorií: torch.Size([12, 200, 4])
Generuji sadu 13/20...
Startovní index: 61, Ex0: [1.4879941e+06 6.3931365e+06 5.3989954e+00
-3.8747860e+01]
INFO: Generátor dat používá hranice X:[1476611.42-1489541.47],
Y:[6384032.63-6400441.34]
Generuji 12 platných trajektorií (metoda zahodení)...
Úspěšně vygenerována trajektorie 1/12 (Pokusů: 1)
Úspěšně vygenerována trajektorie 2/12 (Pokusů: 2)
Úspěšně vygenerována trajektorie 3/12 (Pokusů: 3)
Úspěšně vygenerována trajektorie 4/12 (Pokusů: 9)
Úspěšně vygenerována trajektorie 5/12 (Pokusů: 10)
Úspěšně vygenerována trajektorie 6/12 (Pokusů: 11)
Úspěšně vygenerována trajektorie 7/12 (Pokusů: 14)
Úspěšně vygenerována trajektorie 8/12 (Pokusů: 15)
Úspěšně vygenerována trajektorie 9/12 (Pokusů: 18)
Úspěšně vygenerována trajektorie 10/12 (Pokusů: 20)
Úspěšně vygenerována trajektorie 11/12 (Pokusů: 21)
Úspěšně vygenerována trajektorie 12/12 (Pokusů: 22)

Generování dat dokončeno.
Celkový počet pokusů: 22

Úspěšnost (platné trajektorie / pokusy): 54.55%
Celkový počet vygenerovaných trajektorií: torch.Size([12, 200, 4])
Generuje sadu 14/20...
Startovní index: 191, Ex0: [1.4858399e+06 6.3884220e+06 -1.9870529e+01
-3.0183512e+01]
INFO: Generátor dat používá hranice X:[1476611.42-1489541.47],
Y:[6384032.63-6400441.34]
Generuje 12 platných trajektorií (metoda zahodení)...
Úspěšně vygenerována trajektorie 1/12 (Pokusů: 6)
Úspěšně vygenerována trajektorie 2/12 (Pokusů: 8)
Úspěšně vygenerována trajektorie 3/12 (Pokusů: 31)
Úspěšně vygenerována trajektorie 4/12 (Pokusů: 39)
Úspěšně vygenerována trajektorie 5/12 (Pokusů: 43)
Úspěšně vygenerována trajektorie 6/12 (Pokusů: 44)
Úspěšně vygenerována trajektorie 7/12 (Pokusů: 54)
Úspěšně vygenerována trajektorie 8/12 (Pokusů: 83)
Úspěšně vygenerována trajektorie 9/12 (Pokusů: 89)
Úspěšně vygenerována trajektorie 10/12 (Pokusů: 94)
Úspěšně vygenerována trajektorie 11/12 (Pokusů: 97)
Úspěšně vygenerována trajektorie 12/12 (Pokusů: 101)

Generování dat dokončeno.
Celkový počet pokusů: 101
Úspěšnost (platné trajektorie / pokusy): 11.88%
Celkový počet vygenerovaných trajektorií: torch.Size([12, 200, 4])
Generuje sadu 15/20...
Startovní index: 447, Ex0: [1.4794251e+06 6.3866660e+06 -1.7736534e+01
7.9389377e+00]
INFO: Generátor dat používá hranice X:[1476611.42-1489541.47],
Y:[6384032.63-6400441.34]
Generuje 12 platných trajektorií (metoda zahodení)...
Úspěšně vygenerována trajektorie 1/12 (Pokusů: 1)
Úspěšně vygenerována trajektorie 2/12 (Pokusů: 4)
Úspěšně vygenerována trajektorie 3/12 (Pokusů: 7)
Úspěšně vygenerována trajektorie 4/12 (Pokusů: 8)
Úspěšně vygenerována trajektorie 5/12 (Pokusů: 9)
Úspěšně vygenerována trajektorie 6/12 (Pokusů: 12)
Úspěšně vygenerována trajektorie 7/12 (Pokusů: 14)
Úspěšně vygenerována trajektorie 8/12 (Pokusů: 15)
Úspěšně vygenerována trajektorie 9/12 (Pokusů: 19)
Úspěšně vygenerována trajektorie 10/12 (Pokusů: 23)
Úspěšně vygenerována trajektorie 11/12 (Pokusů: 24)
Úspěšně vygenerována trajektorie 12/12 (Pokusů: 26)

Generování dat dokončeno.
Celkový počet pokusů: 26
Úspěšnost (platné trajektorie / pokusy): 46.15%
Celkový počet vygenerovaných trajektorií: torch.Size([12, 200, 4])

Generuje sadu 16/20...

Startovní index: 476, Ex0: [1.4787792e+06 6.3869160e+06 -2.6363794e+01
-2.1202071e+00]

INFO: Generátor dat používá hranice X:[1476611.42-1489541.47] ,
Y: [6384032.63-6400441.34]

Generuje 12 platných trajektorií (metoda zahodení)...

Úspěšně vygenerována trajektorie 1/12 (Pokusů: 35)
Úspěšně vygenerována trajektorie 2/12 (Pokusů: 44)
Úspěšně vygenerována trajektorie 3/12 (Pokusů: 70)
Úspěšně vygenerována trajektorie 4/12 (Pokusů: 137)
Úspěšně vygenerována trajektorie 5/12 (Pokusů: 171)
Úspěšně vygenerována trajektorie 6/12 (Pokusů: 184)
Úspěšně vygenerována trajektorie 7/12 (Pokusů: 193)
Úspěšně vygenerována trajektorie 8/12 (Pokusů: 201)
Úspěšně vygenerována trajektorie 9/12 (Pokusů: 220)
Úspěšně vygenerována trajektorie 10/12 (Pokusů: 329)
Úspěšně vygenerována trajektorie 11/12 (Pokusů: 342)
Úspěšně vygenerována trajektorie 12/12 (Pokusů: 349)

Generování dat dokončeno.

Celkový počet pokusů: 349

Úspěšnost (platné trajektorie / pokusy): 3.44%

Celkový počet vygenerovaných trajektorií: torch.Size([12, 200, 4])

Generuje sadu 17/20...

Startovní index: 1034, Ex0: [1.4859065e+06 6.3948215e+06 1.1094100e+01
-2.1170458e+01]

INFO: Generátor dat používá hranice X:[1476611.42-1489541.47] ,
Y: [6384032.63-6400441.34]

Generuje 12 platných trajektorií (metoda zahodení)...

Úspěšně vygenerována trajektorie 1/12 (Pokusů: 1)
Úspěšně vygenerována trajektorie 2/12 (Pokusů: 3)
Úspěšně vygenerována trajektorie 3/12 (Pokusů: 4)
Úspěšně vygenerována trajektorie 4/12 (Pokusů: 5)
Úspěšně vygenerována trajektorie 5/12 (Pokusů: 6)
Úspěšně vygenerována trajektorie 6/12 (Pokusů: 7)
Úspěšně vygenerována trajektorie 7/12 (Pokusů: 8)
Úspěšně vygenerována trajektorie 8/12 (Pokusů: 10)
Úspěšně vygenerována trajektorie 9/12 (Pokusů: 11)
Úspěšně vygenerována trajektorie 10/12 (Pokusů: 12)
Úspěšně vygenerována trajektorie 11/12 (Pokusů: 14)
Úspěšně vygenerována trajektorie 12/12 (Pokusů: 16)

Generování dat dokončeno.

Celkový počet pokusů: 16

Úspěšnost (platné trajektorie / pokusy): 75.00%

Celkový počet vygenerovaných trajektorií: torch.Size([12, 200, 4])

Generuje sadu 18/20...

Startovní index: 1232, Ex0: [1.4868870e+06 6.3964760e+06 1.5584729e+00

2.0859661e+01]
INFO: Generátor dat používá hranice X:[1476611.42-1489541.47],
Y: [6384032.63-6400441.34]
Generuje 12 platných trajektorií (metoda zahodení)...
Úspěšně vygenerována trajektorie 1/12 (Pokusů: 2)
Úspěšně vygenerována trajektorie 2/12 (Pokusů: 5)
Úspěšně vygenerována trajektorie 3/12 (Pokusů: 7)
Úspěšně vygenerována trajektorie 4/12 (Pokusů: 9)
Úspěšně vygenerována trajektorie 5/12 (Pokusů: 11)
Úspěšně vygenerována trajektorie 6/12 (Pokusů: 12)
Úspěšně vygenerována trajektorie 7/12 (Pokusů: 13)
Úspěšně vygenerována trajektorie 8/12 (Pokusů: 15)
Úspěšně vygenerována trajektorie 9/12 (Pokusů: 18)
Úspěšně vygenerována trajektorie 10/12 (Pokusů: 26)
Úspěšně vygenerována trajektorie 11/12 (Pokusů: 29)
Úspěšně vygenerována trajektorie 12/12 (Pokusů: 30)

Generování dat dokončeno.
Celkový počet pokusů: 30
Úspěšnost (platné trajektorie / pokusy): 40.00%
Celkový počet vygenerovaných trajektorií: torch.Size([12, 200, 4])
Generuje sadu 19/20...
Startovní index: 54, Ex0: [1.4879459e+06 6.3934035e+06 7.8290997e+00
-3.7658058e+01]
INFO: Generátor dat používá hranice X:[1476611.42-1489541.47],
Y: [6384032.63-6400441.34]
Generuje 12 platných trajektorií (metoda zahodení)...
Úspěšně vygenerována trajektorie 1/12 (Pokusů: 2)
Úspěšně vygenerována trajektorie 2/12 (Pokusů: 3)
Úspěšně vygenerována trajektorie 3/12 (Pokusů: 4)
Úspěšně vygenerována trajektorie 4/12 (Pokusů: 8)
Úspěšně vygenerována trajektorie 5/12 (Pokusů: 12)
Úspěšně vygenerována trajektorie 6/12 (Pokusů: 13)
Úspěšně vygenerována trajektorie 7/12 (Pokusů: 14)
Úspěšně vygenerována trajektorie 8/12 (Pokusů: 16)
Úspěšně vygenerována trajektorie 9/12 (Pokusů: 17)
Úspěšně vygenerována trajektorie 10/12 (Pokusů: 19)
Úspěšně vygenerována trajektorie 11/12 (Pokusů: 21)
Úspěšně vygenerována trajektorie 12/12 (Pokusů: 23)

Generování dat dokončeno.
Celkový počet pokusů: 23
Úspěšnost (platné trajektorie / pokusy): 52.17%
Celkový počet vygenerovaných trajektorií: torch.Size([12, 200, 4])
Generuje sadu 20/20...
Startovní index: 1149, Ex0: [1.4866324e+06 6.3948120e+06 5.0839610e+00
2.2316792e+01]
INFO: Generátor dat používá hranice X:[1476611.42-1489541.47],

Y: [6384032.63-6400441.34]

Generuje 12 platných trajektorií (metoda zahodení)...

Úspěšně vygenerována trajektorie 1/12 (Pokusů: 2)

Úspěšně vygenerována trajektorie 2/12 (Pokusů: 5)

Úspěšně vygenerována trajektorie 3/12 (Pokusů: 8)

Úspěšně vygenerována trajektorie 4/12 (Pokusů: 11)

Úspěšně vygenerována trajektorie 5/12 (Pokusů: 14)

Úspěšně vygenerována trajektorie 6/12 (Pokusů: 15)

Úspěšně vygenerována trajektorie 7/12 (Pokusů: 16)

Úspěšně vygenerována trajektorie 8/12 (Pokusů: 17)

Úspěšně vygenerována trajektorie 9/12 (Pokusů: 19)

Úspěšně vygenerována trajektorie 10/12 (Pokusů: 21)

Úspěšně vygenerována trajektorie 11/12 (Pokusů: 26)

Úspěšně vygenerována trajektorie 12/12 (Pokusů: 27)

Generování dat dokončeno.

Celkový počet pokusů: 27

Úspěšnost (platné trajektorie / pokusy): 44.44%

Celkový počet vygenerovaných trajektorií: torch.Size([12, 200, 4])

Finální trénovací data: x=torch.Size([240, 200, 4]), y=torch.Size([240, 200, 3])

Generuje validační data s náhodnými počátečními podmínkami z GNSS...

Generuje sadu 1/20...

Startovní index: 407, Ex0: [1.4800204e+06 6.3866590e+06 -1.0816915e+01
-3.8981032e+00]

INFO: Generátor dat používá hranice X:[1476611.42-1489541.47],

Y: [6384032.63-6400441.34]

Generuje 1 platných trajektorií (metoda zahodení)...

Úspěšně vygenerována trajektorie 1/1 (Pokusů: 6)

Generování dat dokončeno.

Celkový počet pokusů: 6

Úspěšnost (platné trajektorie / pokusy): 16.67%

Celkový počet vygenerovaných trajektorií: torch.Size([1, 300, 4])

Generuje sadu 2/20...

Startovní index: 1116, Ex0: [1.48665962e+06 6.39420500e+06 1.15104353e+00
1.36823635e+01]

INFO: Generátor dat používá hranice X:[1476611.42-1489541.47],

Y: [6384032.63-6400441.34]

Generuje 1 platných trajektorií (metoda zahodení)...

Úspěšně vygenerována trajektorie 1/1 (Pokusů: 1)

Generování dat dokončeno.

Celkový počet pokusů: 1

Úspěšnost (platné trajektorie / pokusy): 100.00%

Celkový počet vygenerovaných trajektorií: torch.Size([1, 300, 4])

Generuje sadu 3/20...

Startovní index: 859, Ex0: [1.4835009e+06 6.3944540e+06 1.1596151e+01
3.2644203e+01]

```
INFO: Generátor dat používá hranice X:[1476611.42-1489541.47] ,  
Y:[6384032.63-6400441.34]  
Generuje 1 platných trajektorií (metoda zahodení)...  
Úspěšně vygenerována trajektorie 1/1 (Pokusů: 10)  
-----  
Generování dat dokončeno.  
Celkový počet pokusů: 10  
Úspěšnost (platné trajektorie / pokusy): 10.00%  
Celkový počet vygenerovaných trajektorií: torch.Size([1, 300, 4])  
Generuje sadu 4/20...  
Startovní index: 451, Ex0: [ 1.4793568e+06 6.3867070e+06 -1.7254520e+01  
1.2380473e+01]  
INFO: Generátor dat používá hranice X:[1476611.42-1489541.47] ,  
Y:[6384032.63-6400441.34]  
Generuje 1 platných trajektorií (metoda zahodení)...  
Úspěšně vygenerována trajektorie 1/1 (Pokusů: 1)  
-----  
Generování dat dokončeno.  
Celkový počet pokusů: 1  
Úspěšnost (platné trajektorie / pokusy): 100.00%  
Celkový počet vygenerovaných trajektorií: torch.Size([1, 300, 4])  
Generuje sadu 5/20...  
Startovní index: 919, Ex0: [1.4840482e+06 6.3958880e+06 6.1036477e+00  
9.2955494e+00]  
INFO: Generátor dat používá hranice X:[1476611.42-1489541.47] ,  
Y:[6384032.63-6400441.34]  
Generuje 1 platných trajektorií (metoda zahodení)...  
Úspěšně vygenerována trajektorie 1/1 (Pokusů: 1)  
-----  
Generování dat dokončeno.  
Celkový počet pokusů: 1  
Úspěšnost (platné trajektorie / pokusy): 100.00%  
Celkový počet vygenerovaných trajektorií: torch.Size([1, 300, 4])  
Generuje sadu 6/20...  
Startovní index: 1206, Ex0: [ 1.48689062e+06 6.39613750e+06 5.56597449e-02  
-1.15337044e-01]  
INFO: Generátor dat používá hranice X:[1476611.42-1489541.47] ,  
Y:[6384032.63-6400441.34]  
Generuje 1 platných trajektorií (metoda zahodení)...  
Úspěšně vygenerována trajektorie 1/1 (Pokusů: 1)  
-----  
Generování dat dokončeno.  
Celkový počet pokusů: 1  
Úspěšnost (platné trajektorie / pokusy): 100.00%  
Celkový počet vygenerovaných trajektorií: torch.Size([1, 300, 4])  
Generuje sadu 7/20...  
Startovní index: 569, Ex0: [1.4797099e+06 6.3884650e+06 2.4786398e+01  
2.8962617e+00]
```

```
INFO: Generátor dat používá hranice X:[1476611.42-1489541.47] ,  
Y:[6384032.63-6400441.34]  
Generuji 1 platných trajektorií (metoda zahození)...  
Úspěšně vygenerována trajektorie 1/1 (Pokusů: 1)  
-----  
Generování dat dokončeno.  
Celkový počet pokusů: 1  
Úspěšnost (platné trajektorie / pokusy): 100.00%  
Celkový počet vygenerovaných trajektorií: torch.Size([1, 300, 4])  
Generuji sadu 8/20...  
Startovní index: 13, Ex0: [ 1.4876175e+06 6.3949860e+06 6.8650732e+00  
-4.0276253e+01]  
INFO: Generátor dat používá hranice X:[1476611.42-1489541.47] ,  
Y:[6384032.63-6400441.34]  
Generuji 1 platných trajektorií (metoda zahození)...  
Úspěšně vygenerována trajektorie 1/1 (Pokusů: 7)  
-----  
Generování dat dokončeno.  
Celkový počet pokusů: 7  
Úspěšnost (platné trajektorie / pokusy): 14.29%  
Celkový počet vygenerovaných trajektorií: torch.Size([1, 300, 4])  
Generuji sadu 9/20...  
Startovní index: 326, Ex0: [ 1.4821454e+06 6.3868985e+06 -3.9666473e+01  
-2.8664863e+00]  
INFO: Generátor dat používá hranice X:[1476611.42-1489541.47] ,  
Y:[6384032.63-6400441.34]  
Generuji 1 platných trajektorií (metoda zahození)...  
Úspěšně vygenerována trajektorie 1/1 (Pokusů: 270)  
-----  
Generování dat dokončeno.  
Celkový počet pokusů: 270  
Úspěšnost (platné trajektorie / pokusy): 0.37%  
Celkový počet vygenerovaných trajektorií: torch.Size([1, 300, 4])  
Generuji sadu 10/20...  
Startovní index: 865, Ex0: [1.4835720e+06 6.3946490e+06 1.1947921e+01  
3.4164738e+01]  
INFO: Generátor dat používá hranice X:[1476611.42-1489541.47] ,  
Y:[6384032.63-6400441.34]  
Generuji 1 platných trajektorií (metoda zahození)...  
Úspěšně vygenerována trajektorie 1/1 (Pokusů: 5)  
-----  
Generování dat dokončeno.  
Celkový počet pokusů: 5  
Úspěšnost (platné trajektorie / pokusy): 20.00%  
Celkový počet vygenerovaných trajektorií: torch.Size([1, 300, 4])  
Generuji sadu 11/20...  
Startovní index: 696, Ex0: [1.4820844e+06 6.3912220e+06 1.5826292e+01  
2.2250441e+01]
```

```
INFO: Generátor dat používá hranice X:[1476611.42-1489541.47] ,  
Y:[6384032.63-6400441.34]  
Generuji 1 platných trajektorií (metoda zahození)...  
Úspěšně vygenerována trajektorie 1/1 (Pokusů: 1)  
-----  
Generování dat dokončeno.  
Celkový počet pokusů: 1  
Úspěšnost (platné trajektorie / pokusy): 100.00%  
Celkový počet vygenerovaných trajektorií: torch.Size([1, 300, 4])  
Generuji sadu 12/20...  
Startovní index: 569, Ex0: [ 1.4797099e+06 6.3884650e+06 2.4786398e+01  
2.8962617e+00]  
INFO: Generátor dat používá hranice X:[1476611.42-1489541.47] ,  
Y:[6384032.63-6400441.34]  
Generuji 1 platných trajektorií (metoda zahození)...  
Úspěšně vygenerována trajektorie 1/1 (Pokusů: 1)  
-----  
Generování dat dokončeno.  
Celkový počet pokusů: 1  
Úspěšnost (platné trajektorie / pokusy): 100.00%  
Celkový počet vygenerovaných trajektorií: torch.Size([1, 300, 4])  
Generuji sadu 13/20...  
Startovní index: 318, Ex0: [ 1.4824668e+06 6.3869215e+06 -4.0650539e+01  
-2.8372619e+00]  
INFO: Generátor dat používá hranice X:[1476611.42-1489541.47] ,  
Y:[6384032.63-6400441.34]  
Generuji 1 platných trajektorií (metoda zahození)...  
Úspěšně vygenerována trajektorie 1/1 (Pokusů: 13)  
-----  
Generování dat dokončeno.  
Celkový počet pokusů: 13  
Úspěšnost (platné trajektorie / pokusy): 7.69%  
Celkový počet vygenerovaných trajektorií: torch.Size([1, 300, 4])  
Generuji sadu 14/20...  
Startovní index: 440, Ex0: [ 1.4795415e+06 6.3866195e+06 -1.7235598e+01  
5.7602968e+00]  
INFO: Generátor dat používá hranice X:[1476611.42-1489541.47] ,  
Y:[6384032.63-6400441.34]  
Generuji 1 platných trajektorií (metoda zahození)...  
Úspěšně vygenerována trajektorie 1/1 (Pokusů: 2)  
-----  
Generování dat dokončeno.  
Celkový počet pokusů: 2  
Úspěšnost (platné trajektorie / pokusy): 50.00%  
Celkový počet vygenerovaných trajektorií: torch.Size([1, 300, 4])  
Generuji sadu 15/20...  
Startovní index: 689, Ex0: [1.4819522e+06 6.3910575e+06 2.2486538e+01  
2.4028906e+01]
```

```
INFO: Generátor dat používá hranice X:[1476611.42-1489541.47] ,  
Y:[6384032.63-6400441.34]  
Generuji 1 platných trajektorií (metoda zahození)...  
Úspěšně vygenerována trajektorie 1/1 (Pokusů: 1)  
-----  
Generování dat dokončeno.  
Celkový počet pokusů: 1  
Úspěšnost (platné trajektorie / pokusy): 100.00%  
Celkový počet vygenerovaných trajektorií: torch.Size([1, 300, 4])  
Generuji sadu 16/20...  
Startovní index: 209, Ex0: [ 1.4855200e+06 6.3879445e+06 -1.2059240e+01  
-1.6939651e+01]  
INFO: Generátor dat používá hranice X:[1476611.42-1489541.47] ,  
Y:[6384032.63-6400441.34]  
Generuji 1 platných trajektorií (metoda zahození)...  
Úspěšně vygenerována trajektorie 1/1 (Pokusů: 2)  
-----  
Generování dat dokončeno.  
Celkový počet pokusů: 2  
Úspěšnost (platné trajektorie / pokusy): 50.00%  
Celkový počet vygenerovaných trajektorií: torch.Size([1, 300, 4])  
Generuji sadu 17/20...  
Startovní index: 189, Ex0: [ 1.4858800e+06 6.3884830e+06 -2.0166639e+01  
-3.0556942e+01]  
INFO: Generátor dat používá hranice X:[1476611.42-1489541.47] ,  
Y:[6384032.63-6400441.34]  
Generuji 1 platných trajektorií (metoda zahození)...  
Úspěšně vygenerována trajektorie 1/1 (Pokusů: 7)  
-----  
Generování dat dokončeno.  
Celkový počet pokusů: 7  
Úspěšnost (platné trajektorie / pokusy): 14.29%  
Celkový počet vygenerovaných trajektorií: torch.Size([1, 300, 4])  
Generuji sadu 18/20...  
Startovní index: 778, Ex0: [1.4828445e+06 6.3927340e+06 1.7633008e+00  
7.5145102e+00]  
INFO: Generátor dat používá hranice X:[1476611.42-1489541.47] ,  
Y:[6384032.63-6400441.34]  
Generuji 1 platných trajektorií (metoda zahození)...  
Úspěšně vygenerována trajektorie 1/1 (Pokusů: 1)  
-----  
Generování dat dokončeno.  
Celkový počet pokusů: 1  
Úspěšnost (platné trajektorie / pokusy): 100.00%  
Celkový počet vygenerovaných trajektorií: torch.Size([1, 300, 4])  
Generuji sadu 19/20...  
Startovní index: 198, Ex0: [ 1.4857030e+06 6.3882145e+06 -1.9574419e+01  
-2.7659798e+01]
```

```

INFO: Generátor dat používá hranice X:[1476611.42-1489541.47] ,
Y:[6384032.63-6400441.34]
Generuje 1 platných trajektorií (metoda zahodení)...
Úspěšně vygenerována trajektorie 1/1 (Pokusů: 15)
-----
Generování dat dokončeno.
Celkový počet pokusů: 15
Úspěšnost (platné trajektorie / pokusy): 6.67%
Celkový počet vygenerovaných trajektorií: torch.Size([1, 300, 4])
Generuje sadu 20/20...
Startovní index: 735, Ex0: [1.4826289e+06 6.3919320e+06 1.3488583e+01
1.7407158e+01]
INFO: Generátor dat používá hranice X:[1476611.42-1489541.47] ,
Y:[6384032.63-6400441.34]
Generuje 1 platných trajektorií (metoda zahodení)...
Úspěšně vygenerována trajektorie 1/1 (Pokusů: 3)
-----
Generování dat dokončeno.
Celkový počet pokusů: 3
Úspěšnost (platné trajektorie / pokusy): 33.33%
Celkový počet vygenerovaných trajektorií: torch.Size([1, 300, 4])
Finální validační data: x=torch.Size([20, 300, 4]), y=torch.Size([20, 300, 3])

```

DataLoadery jsou připraveny pro trénink.

```

[7]: # #_
# =====#
# # 1. KONFIGURACE TESTU
# #_
# =====#
# TEST_SEQ_LEN = 500 # Změňte zpět na 100 nebo kolik potřebujete
# NUM_TEST_TRAJ = 3
# J_SAMPLES_TEST = 25

# #_
# =====#
# # 2. PŘÍPRAVA DAT (OPRAVENO)
# #_
# =====#
# print(f"\nGeneruje {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_for_map(
#     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]

```

[8]:

```

# 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

# # Nastavení seedu pro reprodukovatelnost tohoto běhu
# torch.manual_seed(42)
# np.random.seed(42)
# random.seed(42)
# state_knet = StateKalmanNet(system_model, device=device,
#                               hidden_size_multiplier=12).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=40,
#     clip_grad=1.0
# )

```

[9]:

```

import torch
import torch.nn as nn
from torch.utils.data import TensorDataset, DataLoader
import numpy as np
import os
import random
import csv

```

```

from datetime import datetime
import pandas as pd
from copy import deepcopy
from state_NN_models import StateKalmanNet_v2_4D_tan

# Nastavení seedu pro reproducovatelnost tohoto běhu
torch.manual_seed(42)
np.random.seed(42)
random.seed(42)
state_knet = StateKalmanNet_v2_4D_tan(system_model, device=device,
                                         hidden_size_multiplier=12, output_layer_multiplier=4, num_gru_layers=1).
                                         .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=30,
    clip_grad=10.0
)

```

```

StateKalmanNet_v2_4D_tan(
    (dnn): DNN_KalmanNet_v2(
        (input_norm): LayerNorm((14,), eps=1e-05, elementwise_affine=True)
        (input_layer): Sequential(
            (0): Linear(in_features=14, out_features=672, bias=True)
            (1): ReLU()
        )
        (gru): GRU(672, 250)
        (output_hidden_layer): Sequential(
            (0): Linear(in_features=250, out_features=48, bias=True)
            (1): ReLU()
        )
        (output_final_linear): Linear(in_features=48, out_features=12, bias=True)
    )
)
/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() "
INFO: Detekováno z atributu modelu, že vrací kovarianci: False
Epoch [5/200], Train Loss: 687281.793750, Val Loss: 1041529.750000
Epoch [10/200], Train Loss: 44363.153654, Val Loss: 590109.829427

```

```
Epoch [15/200], Train Loss: 15917.549062, Val Loss: 368235.541667
Epoch [20/200], Train Loss: 17426.241077, Val Loss: 353564.708333
Epoch [25/200], Train Loss: 17089.090741, Val Loss: 359826.812500
Epoch [30/200], Train Loss: 16971.360282, Val Loss: 362062.270833
Epoch [35/200], Train Loss: 17047.533054, Val Loss: 363697.562500
Epoch [40/200], Train Loss: 17067.666323, Val Loss: 365474.010417
```

Early stopping spuštěno po 41 epochách.

Trénování dokončeno.

Načítám nejlepší model s validační chybou: 260455.092855

```
[9]: StateKalmanNet_v2_4D_tan(
    (dnn): DNN_KalmanNet_v2(
        (input_norm): LayerNorm((14,), eps=1e-05, elementwise_affine=True)
        (input_layer): Sequential(
            (0): Linear(in_features=14, out_features=672, bias=True)
            (1): ReLU()
        )
        (gru): GRU(672, 250)
        (output_hidden_layer): Sequential(
            (0): Linear(in_features=250, out_features=48, bias=True)
            (1): ReLU()
        )
        (output_final_linear): Linear(in_features=48, out_features=12, bias=True)
    )
)
```

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

# model_config_phase1 = {
#     "hidden_size_multiplier": 12,
#     "output_layer_multiplier": 4,
#     "num_gru_layers": 1,
#     "init_min_dropout": 0.5,
#     "init_max_dropout": 0.8
# }
```

```

# train_config_phase1 = {
#     "total_train_iter": 1000,
#     "learning_rate": 1e-5,
#     "clip_grad": 1.0,
#     "J_samples": 10,
#     "validation_period": 20,
#     "logging_period": 2000,
#     "warmup_iterations": 100 # Trénuj prvních 400 iterací jen na MSE
# }

# # Vytvoření modelu
# state_bkn_knet = StateBayesianKalmanNet(
#     system_model,
#     device=device,
#     **model_config_phase1
# ).to(device)

# # Spuštění tréninku
# # Používáme `run_training_session`, která vrací slovník s výsledky
# results = trainer.
#     ↪run_training_session_trajectory_with_gaussian_nll_training_fcn(model=state_bkn_knet,
#     train_loader=train_loader,
#     val_loader=val_loader,
#     device=device,
#     **train_config_phase1
# )

# 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)

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

```

[15]:

```

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 = 5
J_SAMPLES_TEST = 25

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

# Nyní předáme oríznutou sekvenci 'u', takže i 'x' a 'y' budou mít správnou\n
# délku.
x_test, y_test = utils.generate_data_for_map(
    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_knet_diagnostics_cpu = []
print(f"\nVyhodnocuji modely na {NUM_TEST_TRAJ} testovacích trajektoriích...")

state_knet.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)

        # # --- 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)

```

```

# # --- B. Klasický StateKalmanNet (pouze MSE) ---
# state_knet_F3.reset(batch_size=1, initial_state=initial_state)
# classic_knet_preds_F3 = []
# for t in range(1, TEST_SEQ_LEN):
#     x_filtered_t = state_knet_F3.step(y_test_seq_gpu[t, :].
˓→unsqueeze(0))
#         classic_knet_preds_F3.append(x_filtered_t)
# full_x_hat_classic_knet_F3 = torch.cat([initial_state, torch.
˓→cat(classic_knet_preds_F3, dim=0)], dim=0)

ukf_i_res = ukf_ideal.process_sequence(
    y_seq=y_test_seq_gpu,
    Ex0=system_model.Ex0,
    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=system_model.Ex0, 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_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())
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_knet_F3 = []

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

with torch.no_grad():
    for i in range(NUM_TEST_TRAJ):
        x_true = all_x_true_cpu[i]
        def get_metrics(x_hat_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 délka! 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)
            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_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"{'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"{'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)

```

Generuje 5 testovacích trajektorií o délce 800...

INFO: Generátor dat používá hranice X:[1476611.42-1489541.47],
Y:[6384032.63-6400441.34]

Generuje 5 platných trajektorií (metoda zahodení)...

Úspěšně vygenerována trajektorie 1/5 (Pokusů: 103)

Úspěšně vygenerována trajektorie 2/5 (Pokusů: 309)

Úspěšně vygenerována trajektorie 3/5 (Pokusů: 374)

Úspěšně vygenerována trajektorie 4/5 (Pokusů: 716)

Úspěšně vygenerována trajektorie 5/5 (Pokusů: 807)

Generování dat dokončeno.

Celkový počet pokusů: 807

Úspěšnost (platné trajektorie / pokusy): 0.62%
Celkový počet vygenerovaných trajektorií: torch.Size([5, 800, 4])
Generování dat dokončeno.
y shape: torch.Size([5, 800, 3])
x shape: torch.Size([5, 800, 4])

Vyhodnocuji modely na 5 testovacích trajektoriích...
PF-SIR (ideální model) dokončen pro trajektorii 1/5.
Dokončena trajektorie 1/5...
PF-SIR (ideální model) dokončen pro trajektorii 2/5.
Dokončena trajektorie 2/5...
PF-SIR (ideální model) dokončen pro trajektorii 3/5.
Dokončena trajektorie 3/5...
PF-SIR (ideální model) dokončen pro trajektorii 4/5.
Dokončena trajektorie 4/5...
PF-SIR (ideální model) dokončen pro trajektorii 5/5.
Dokončena trajektorie 5/5...

Počítám finální metriky pro jednotlivé trajektorie...

=====

trajektorie: 1/5

=====

KNet (pouze MSE)	136613.1406	N/A
UKF (Ideální model)	119.0470	8.9094
PF-SIR (Ideální model)	21.8257	3.3662

=====

=====

trajektorie: 2/5

=====

KNet (pouze MSE)	5770.0850	N/A
UKF (Ideální model)	5537.2783	195.1413
PF-SIR (Ideální model)	41.2591	4.8065

=====

=====

trajektorie: 3/5

=====

KNet (pouze MSE)	156940.2656	N/A
UKF (Ideální model)	40.7404	4.4464
PF-SIR (Ideální model)	1770.7529	4.6807

=====

trajektorie: 4/5

KNet (pouze MSE)	50767.4531	N/A
UKF (Ideální model)	133.0060	7.9229
PF-SIR (Ideální model)	201.8086	9.5722

trajektorie: 5/5

KNet (pouze MSE)	208758.6406	N/A
UKF (Ideální model)	32.2030	4.1538
PF-SIR (Ideální model)	26.1343	4.0436

FINÁLNÍ VÝSLEDKY (průměr přes 5 běhů)

Model	Průměrné MSE	Průměrný ANEES
---	---	---
--- Model-Based Filters ---		
KNet (pouze MSE)	111769.9170	N/A
---	---	---
--- Benchmarks ---		
UKF (Ideální model)	1172.4549	44.1148
PF-SIR (Ideální model)	412.3561	5.2938

```
[16]: import matplotlib.pyplot as plt
import numpy as np
import torch # Přidáno pro výpočet chyby a práci s tenzory

# <<< ZDE PŘEDPOKLÁDÁME EXISTENCI NÁSLEDUJÍCÍCH PROMĚNNÝCH >>>
# NUM_TEST_TRAJ = ... # Počet testovacích trajektorií
# all_x_true_cpu = [...] # Seznam pravdivých stavů
# all_x_hat_classic_knet_cpu = [...] # Seznam odhadů KNet
# all_x_hat_pf_sir_ideal_cpu = [...] # Seznam odhadů PF-SIR
# all_knet_diagnostics_cpu = [...] # <-- PŘEDPOKLAD: Seznam slovníků z
    ↵vyhodnocení
# <<< KONEC PŘEDPOKLADŮ >>>

map_bounds = {
    'x_min': 1476611.42,
    'x_max': 1489541.47,
```

```

'y_min': 6384032.63,
'y_max': 6400441.34
}

# Zvolíme trajektorii pro vykreslení
index = 2 # Vykreslíme poslední trajektorii
if index < 0: index = 0

# =====
# 1. PŘÍPRAVA DAT (stejná jako předtím + NOVÁ DIAGNOSTIKA)
# =====

# --- Získání diagnostického slovníku pro daný 'index' ---
try:
    knet_diagnostics = all_knet_diagnostics_cpu[index]
    plot_diagnostics = True
except (NameError, IndexError):
    print("Varování: 'all_knet_diagnostics_cpu' nenalezeno nebo je prázdné. ↴\nDiagnostické grafy (K, h, inovace) nebudou vykresleny.")
    plot_diagnostics = False
    plot_gains = False # Vypneme i graf zisku K

# --- Příprava dat trajektorií ---
x_true_plot = all_x_true_cpu[index].numpy()
x_true_tensor = all_x_true_cpu[index]
x_knet_tensor = all_x_hat_classic_knet_cpu[index]
x_pf_tensor = all_x_hat_pf_sir_ideal_cpu[index]

# --- Výpočet RMSE ---
squared_error = (x_knet_tensor - x_true_tensor)**2
rmse_per_step = torch.sqrt(squared_error).numpy()

num_steps = x_true_plot.shape[0]
time_axis = np.arange(num_steps) # Časová osa od 0 do T-1
gain_time_axis = np.arange(1, num_steps) # Časová osa od 1 do T-1

# --- Příprava dat pro diagnostiku (pokud existují) ---
if plot_diagnostics:
    # --- Kalmanův zisk (K_t) ---
    try:
        kalman_gains_history = knet_diagnostics['K_history']
        # K[0, :, 0] znamená [batch_index, vsechny_stavy, první_prvek_merení ↴\n(vyska)]
        gains_col0_cpu = [K[0, :, 0].cpu().numpy() for K in ↴\nkalman_gains_history]
        gains_col0_np = np.array(gains_col0_cpu) # Převede seznam polí na 2D ↴\npole [čas (T-1), prvek (4)]
    
```

```

plot_gains = True

if gains_col0_np.shape[0] != len(gain_time_axis):
    print(f"Varování: Délka historie zisku ({gains_col0_np.shape[0]}))")
    ↵neodpovídá časové ose ({len(gain_time_axis)}). Graf zisku K nebude vykreslen.
    ↵")
    plot_gains = False

except Exception as e:
    print(f"Nastala chyba při zpracování Kalmanova zisku: {e}. Grafy K")
    ↵nebudou vykresleny.")
    plot_gains = False

# --- Skrytý stav (h_t) ---
try:
    h_history = knet_diagnostics['h_history']
    # h_history má T prvků (h_0 až h_T-1), h_t má tvar [num_layers, batch, ↵
    ↵hidden_dim]
    # Spočítáme L2 normu pro každý krok
    h_norms = [torch.norm(h.squeeze(1)).item() for h in h_history] # ↵
    ↵Squeeze pro odstranění 'batch' dim
    plot_h_norm = True

    if len(h_norms) != len(time_axis):
        print(f"Varování: Délka historie stavu h ({len(h_norms)})")
        ↵neodpovídá časové ose ({len(time_axis)}). Graf h nebude vykreslen.")
        plot_h_norm = False

except Exception as e:
    print(f"Nastala chyba při zpracování skrytého stavu h: {e}. Graf h")
    ↵nebudou vykreslen.")
    plot_h_norm = False

# --- Inovace (Delta y_t) ---
try:
    innovation_history = knet_diagnostics['innovation_history']
    # innovation_history má T-1 prvků (pro t=1 až T-1), inovace má tvar ↵
    ↵[batch, obs_dim]
    innov_norms = [torch.norm(innov.squeeze(0)).item() for innov in ↵
    ↵innovation_history] # Squeeze pro odstranění 'batch' dim
    plot_innov_norm = True

    if len(innov_norms) != len(gain_time_axis):
        print(f"Varování: Délka historie inovace ({len(innov_norms)})")
        ↵neodpovídá časové ose ({len(gain_time_axis)}). Graf inovace nebude vykreslen.
        ↵")

```

```

    plot_innov_norm = False

except Exception as e:
    print(f"Nastala chyba při zpracování inovace: {e}. Graf inovace nebude vykreslen.")
    plot_innov_norm = False


# --- Popisky grafů (Labels) ---
state_labels = [
    'Pozice X [m]',
    'Pozice Y [m]',
    'Rychlost vX [m/s]',
    'Rychlost vY [m/s]'
]
error_labels = [
    'RMSE Pozice X [m]',
    'RMSE Pozice Y [m]',
    'RMSE Rychlost vX [m/s]',
    'RMSE Rychlost vY [m/s]'
]
gain_labels = [
    'K[0,0] (Výška -> Pozice X)',
    'K[1,0] (Výška -> Pozice Y)',
    'K[2,0] (Výška -> Rychlost vX)',
    'K[3,0] (Výška -> Rychlost vY)'
]
diagnostic_labels = {
    'h_norm': 'L2 Norma skrytého stavu $h_t$',
    'innov_norm': 'L2 Norma inovace $\Delta y_t$'
}

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

# --- Graf 1: Trajektorie ---
fig1, axes1 = plt.subplots(4, 1, figsize=(12, 14), sharex=True)
fig1.suptitle(f'Detailní porovnání odhadů stavu v čase (Trajektorie {index+1})', fontsize=16)

# --- Graf 2: RMSE KNet ---
fig2, axes2 = plt.subplots(4, 1, figsize=(12, 14), sharex=True)
fig2.suptitle(f'RMSE odhadu pro jednotlivé složky stavu v čase (KNet, Traj.{index+1})', fontsize=16)

```

```

# --- Graf 3: Kalmanův zisk KNet ---
if plot_gains:
    fig3, axes3 = plt.subplots(4, 1, figsize=(12, 14), sharex=True)
    fig3.suptitle(f'Vývoj prvků 1. sloupce Kalmanova zisku KNet v čase (Traj. {index+1})', fontsize=16)

# --- Graf 4: Diagnostika (h_t a Inovace) ---
if plot_diagnostics and plot_h_norm and plot_innov_norm:
    fig4, axes4 = plt.subplots(2, 1, figsize=(12, 10), sharex=True)
    fig4.suptitle(f'Diagnostika vnitřních stavů KNet v čase (Traj. {index+1})', fontsize=16)
else:
    plot_h_norm = False
    plot_innov_norm = False

# --- Smyčka přes všechny 4 složky stavu (pro fig1, fig2, fig3) ---
for i in range(4):
    # --- Graf 1: Trajektorie ---
    ax1 = axes1[i]
    ax1.plot(time_axis, x_true_plot[:, i], 'r-', linewidth=2.0, label='Referenční hodnota')
    ax1.plot(time_axis, x_knet_tensor[:, i].numpy(), 'g--', linewidth=1.5, label='Odhad KNet')
    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)
    ax1.legend()

    if i == 0: # Graf pro Pozici X
        ax1.axhline(map_bounds['x_min'], color='grey', linestyle=':', linewidth=1.5, label='Hranice mapy X')
        ax1.axhline(map_bounds['x_max'], color='grey', linestyle=':', linewidth=1.5)
        print(f"INFO: Přidávám hranice X ({map_bounds['x_min']:.2f}, {map_bounds['x_max']:.2f}) do grafu Pozice X.")
    elif i == 1: # Graf pro Pozici Y
        ax1.axhline(map_bounds['y_min'], color='grey', linestyle=':', linewidth=1.5, label='Hranice mapy Y')
        ax1.axhline(map_bounds['y_max'], color='grey', linestyle=':', linewidth=1.5)
        print(f"INFO: Přidávám hranice Y ({map_bounds['y_min']:.2f}, {map_bounds['y_max']:.2f}) do grafu Pozice Y.")

# --- Graf 2: Chyba (RMSE) ---

```

```

ax2 = axes2[i]
ax2.plot(time_axis, rmse_per_step[:, i], 'b-', linewidth=1.5, label=f'RMSE_{i+1}_KNet (Avg: {np.mean(rmse_per_step[1:, i]):.2f})')
ax2.set_ylabel(error_labels[i])
ax2.grid(True)
ax2.legend()

# --- Graf 3: Kalmanův zisk (pokud jsou data k dispozici) ---
if plot_gains:
    ax3 = axes3[i]
    ax3.plot(gain_time_axis, gains_col0_np[:, i], 'k-', linewidth=1.5, label=f'{gain_labels[i]} (Avg: {np.mean(gains_col0_np[:, i]):.4f})')
    ax3.set_ylabel(gain_labels[i])
    ax3.grid(True)
    ax3.legend()

# --- Vykreslení Grafu 4 (Diagnostika) ---
if plot_h_norm:
    ax4_h = axes4[0]
    ax4_h.plot(time_axis, h_norms, 'darkorange', linewidth=1.5, label=f'Norma ${h_t}$')
    ax4_h.set_ylabel(diagnostic_labels['h_norm'])
    ax4_h.grid(True)
    ax4_h.legend()
    # Povolíme logaritmickou osu Y, pokud jsou hodnoty velmi odlišné
    ax4_h.set_yscale('log')

if plot_innov_norm:
    ax4_innov = axes4[1]
    ax4_innov.plot(gain_time_axis, innov_norms, 'purple', linewidth=1.5, label=f'Norma $\Delta y_t$')
    ax4_innov.set_ylabel(diagnostic_labels['innov_norm'])
    ax4_innov.set_xlabel('Časový krok [s]') # Popisek osy X jen u spodního grafu
    ax4_innov.grid(True)
    ax4_innov.legend()
    ax4_innov.set_yscale('log') # Inovace může také explodovat

# Nastavení popisků pro sdílené osy X
axes1[-1].set_xlabel('Časový krok [s]')
axes2[-1].set_xlabel('Časový krok [s]')
if plot_gains:
    axes3[-1].set_xlabel('Časový krok [s]')

# Zlepšíme rozložení
fig1.tight_layout(rect=[0, 0.03, 1, 0.96])
fig2.tight_layout(rect=[0, 0.03, 1, 0.96])
if plot_gains:

```

```

    fig3.tight_layout(rect=[0, 0.03, 1, 0.96])
if plot_h_norm or plot_innov_norm:
    fig4.tight_layout(rect=[0, 0.03, 1, 0.96])

plt.show()

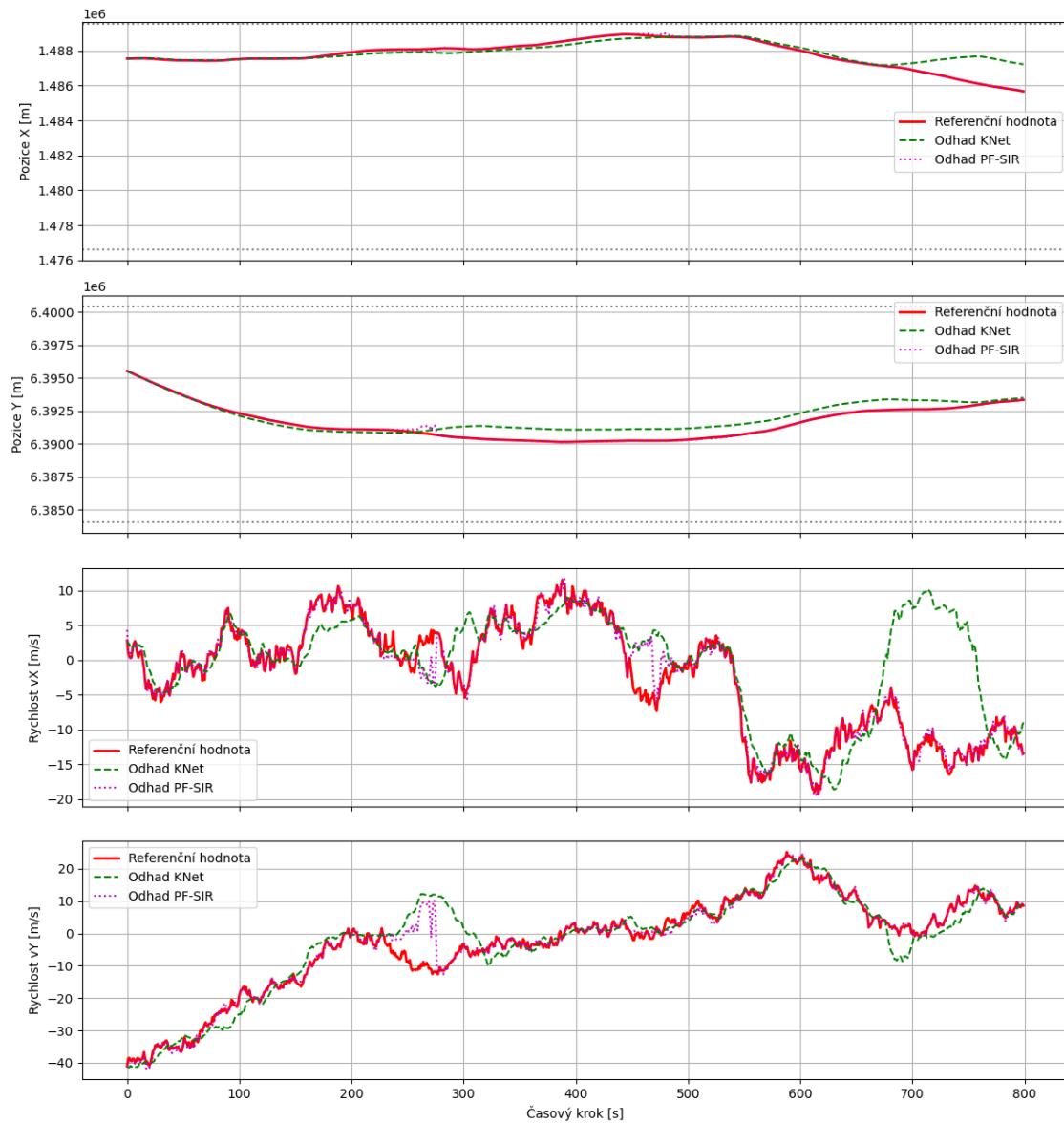
```

Vykresluji grafy...

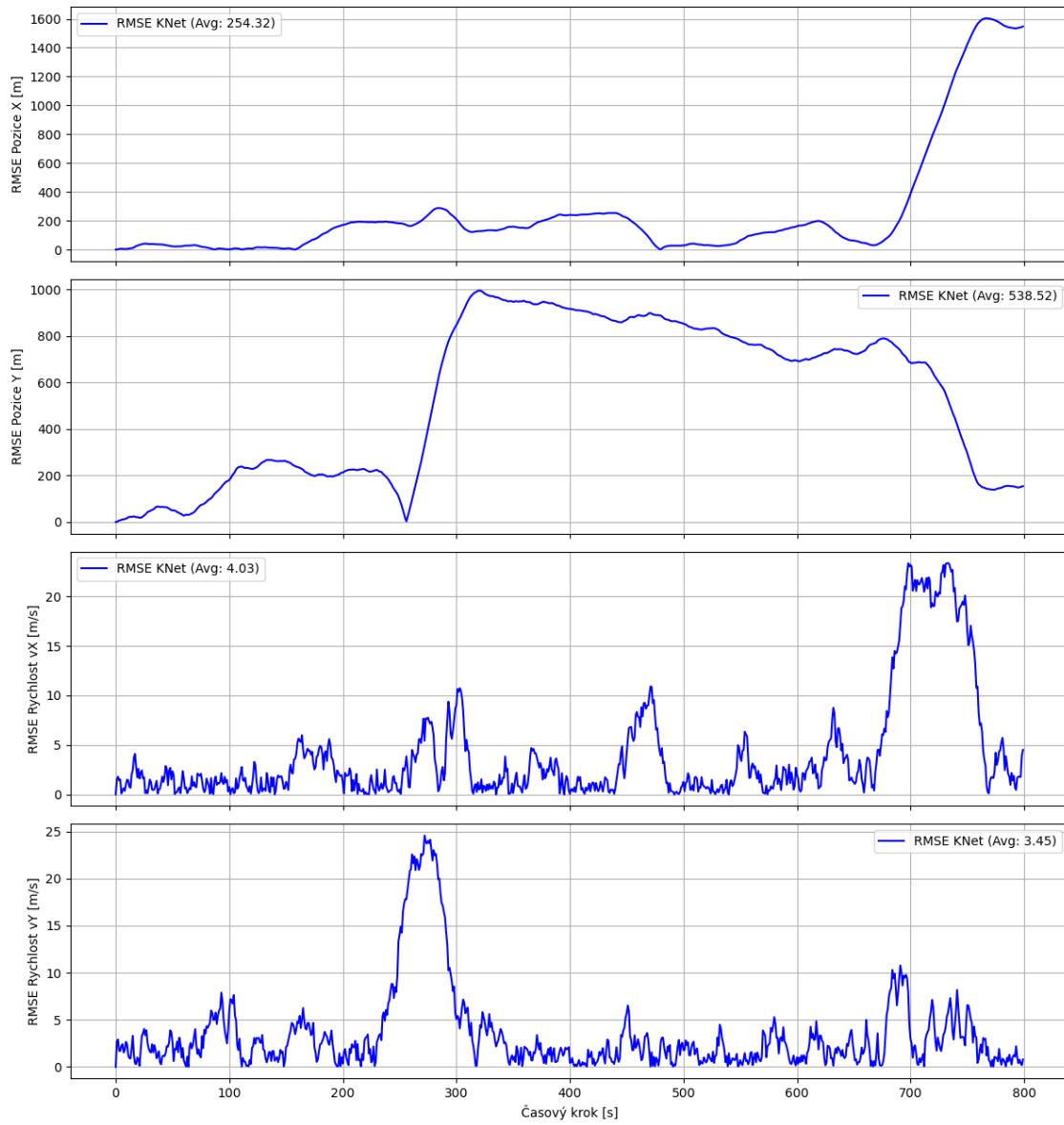
INFO: Přidávám hranice X (1476611.42, 1489541.47) do grafu Pozice X.

INFO: Přidávám hranice Y (6384032.63, 6400441.34) do grafu Pozice Y.

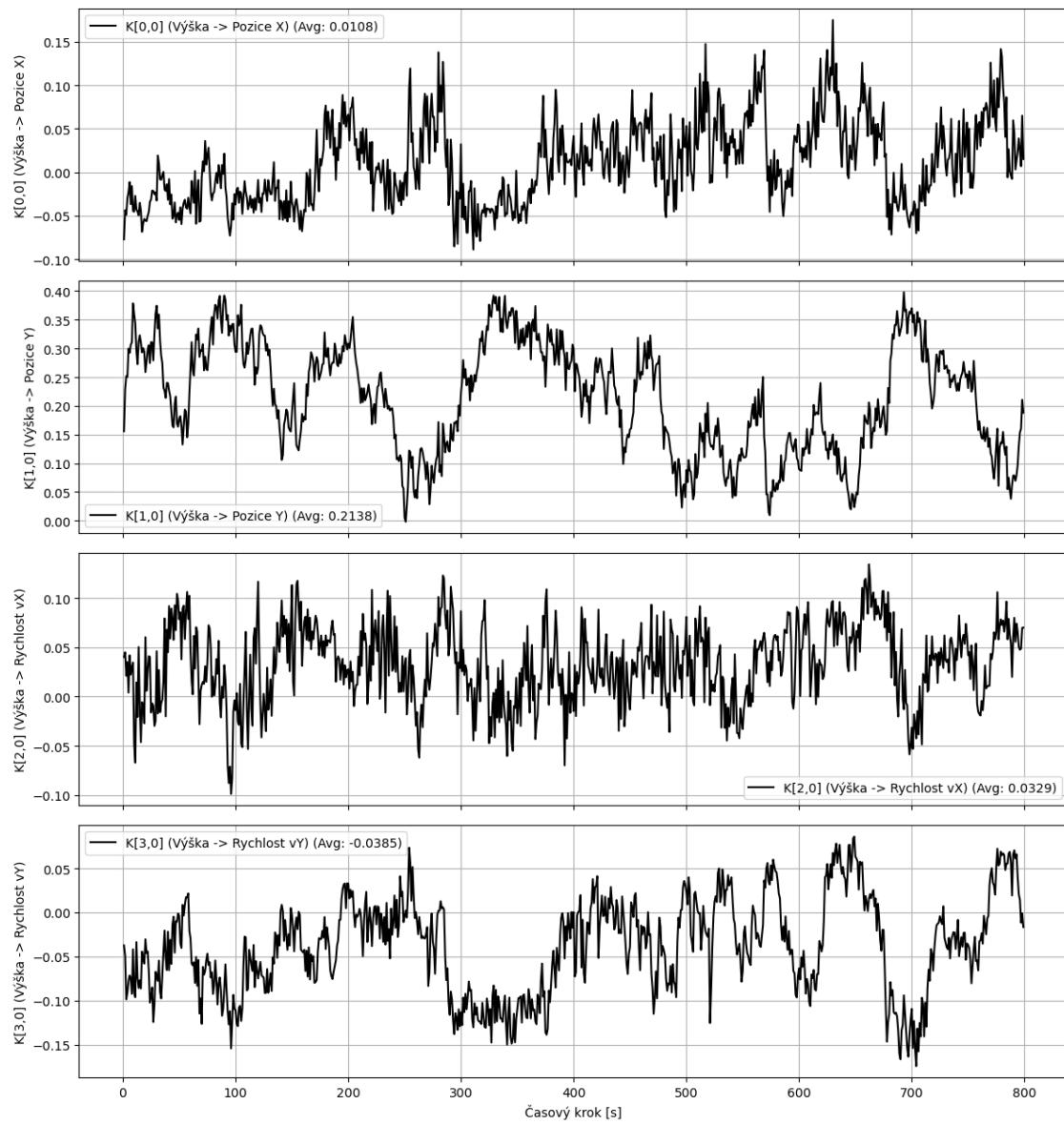
Detailní porovnání odhadů stavu v čase (Trajektorie 3)



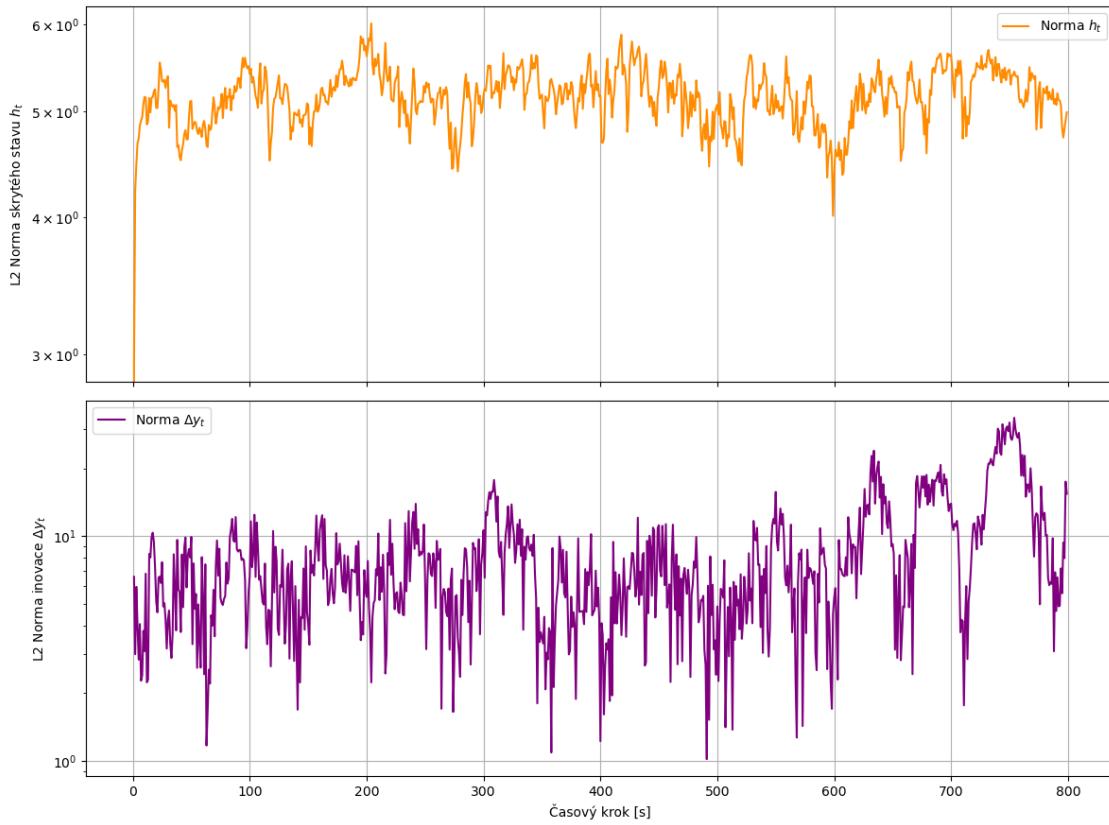
RMSE odhadu pro jednotlivé složky stavu v čase (KNet, Traj. 3)



Vývoj prvků 1. sloupce Kalmanova zisku KNet v čase (Traj. 3)



Diagnostika vnitřních stavů KNet v čase (Traj. 3)



```
[13]: # 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()

```

```

[14]: # 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 trajektorii
# # full_x_hat_classic_knet: Tenzor s odhadu z klasického KNetu
# # full_x_hat_bkn: Tenzor s odhadu 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 trajectorie')

# # --- 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()

```