BKN_training_checkpoint

February 10, 2026

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

# Robust path finding for data.mat
current_path = Path.cwd()
possible_data_paths = [
    current_path / 'data' / 'data.mat',
    current_path.parent / 'data' / 'data.mat',
    current_path.parent.parent / 'data' / 'data.mat',
    # Fallback absolute path
    Path('/home/luky/skola/KalmanNet-for-state-estimation/data/data.mat')
]

dataset_path = None
for p in possible_data_paths:
    if p.exists():
        dataset_path = p
        break

if dataset_path is None or not dataset_path.exists():
    print("Warning: data.mat not found automatically.")
    dataset_path = Path('data/data.mat')

print(f"Dataset path: {dataset_path}")

# Add project root to sys.path (2 levels up from debug/test)
notebook_dir = os.getcwd()
project_root = os.path.abspath(os.path.join(notebook_dir, '..', '..'))
if project_root not in sys.path:
    sys.path.insert(0, project_root)
print(f"Project root added: {project_root}")

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

```
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)
if torch.cuda.is_available():
    torch.cuda.manual_seed_all(42)

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

```
device: cpu
```

```
/home/luky/.local/lib/python3.10/site-packages/torch/cuda/__init__.py:118:
UserWarning: CUDA initialization: CUDA unknown error - this may be due to an
incorrectly set up environment, e.g. changing env variable CUDA_VISIBLE_DEVICES
after program start. Setting the available devices to be zero. (Triggered
internally at ../c10/cuda/CUDAFunctions.cpp:108.)
  return torch._C._cuda_getDeviceCount() > 0
```

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

print(f"Dimensions of 1D X axis: {x_axis_unique.shape}")
print(f"Dimensions of 1D Y axis: {y_axis_unique.shape}")
print(f"Dimensions of 2D elevation data Z: {souradniceZ_mapa.shape}")

terMap_interpolator = RegularGridInterpolator(
    (y_axis_unique, x_axis_unique),
```

```
    souradniceZ_mapa,
    bounds_error=False,
    fill_value=np.nan
)

def terMap(px, py):
    # Query bilinear interpolation over the terrain map
    points_to_query = np.column_stack((py, px))
    return terMap_interpolator(points_to_query)
```

```
Dimensions of 1D X axis: (2500,)
Dimensions of 1D Y axis: (2500,)
Dimensions of 2D elevation data Z: (2500, 2500)
```

[4]:
```
import torch
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[:2, 1] - souradniceGNSS[:2, 0]
# initial_velocity_np = torch.from_numpy()
initial_velocity = torch.from_numpy(np.array([0,0]))

initial_position = torch.from_numpy(souradniceGNSS[:2, 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]])
import torch.nn.functional as func

def h_nl_differentiable(x: torch.Tensor, map_tensor, x_min, x_max, y_min,␣
 ↪y_max) -> torch.Tensor:
    batch_size = x.shape[0]

    px = x[:, 0]
    py = x[:, 1]

    px_norm = 2.0 * (px - x_min) / (x_max - x_min) - 1.0
    py_norm = 2.0 * (py - y_min) / (y_max - y_min) - 1.0

    sampling_grid = torch.stack((px_norm, py_norm), dim=1).view(batch_size, 1,␣
 ↪1, 2)

    vyska_terenu_batch = func.grid_sample(
        map_tensor.expand(batch_size, -1, -1, -1),
        sampling_grid,
        mode='bilinear',
        padding_mode='border',
        align_corners=True
    )

    vyska_terenu = vyska_terenu_batch.view(batch_size)

    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)

    return result

x_axis_unique = souradniceX_mapa[0, :]
y_axis_unique = souradniceY_mapa[:, 0]
terMap_tensor = torch.from_numpy(souradniceZ_mapa).float().unsqueeze(0).
 ↪unsqueeze(0).to(device)
x_min, x_max = x_axis_unique.min(), x_axis_unique.max()
y_min, y_max = y_axis_unique.min(), y_axis_unique.max()
```

```
h_wrapper = lambda x: h_nl_differentiable(
    x,
    map_tensor=terMap_tensor,
    x_min=x_min,
    x_max=x_max,
    y_min=y_min,
    y_max=y_max
)

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_wrapper,
    x_axis_unique=x_axis_unique,
    y_axis_unique=y_axis_unique,
    device=device
)
```

```
tensor([1487547.1250, 6395520.5000,       0.0000,       0.0000])
INFO: DynamicSystemTAN inicializován s hranicemi mapy:
  X: [1476611.42, 1489541.47]
  Y: [6384032.63, 6400441.34]
```

[5]:
```
import torch
from torch.utils.data import TensorDataset, DataLoader
from utils import utils
import torch
import numpy as np
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 TAN
from utils import trainer

torch.manual_seed(42)
np.random.seed(42)
random.seed(42)
```

[6]:
```
import torch
from torch.utils.data import TensorDataset, DataLoader
import os
from utils import trainer # Předpokládám, že toto máš

# === 1. ZJEDNODUŠENÝ DATA MANAGER (BEZ NORMALIZACE) ===
class NavigationDataManager:
    def __init__(self, data_dir):
        """
        Jen držák na cestu k datům. Žádná statistika, žádná normalizace.
        """
        self.data_dir = data_dir

    def get_dataloader(self, seq_len, split='train', shuffle=True,␣
 ↪batch_size=32):
        # Sestavení cesty: ./generated_data/len_100/train.pt
        path = os.path.join(self.data_dir, f'len_{seq_len}', f'{split}.pt')

        if not os.path.exists(path):
            raise FileNotFoundError(f" Dataset nenalezen: {path}")

        # Načtení tenzorů
        data = torch.load(path)
        x = data['x'] # Stav [Batch, Seq, DimX]
        y = data['y'] # Měření [Batch, Seq, DimY] - RAW DATA

        # Vytvoření datasetu
        dataset = TensorDataset(x, y)

        return DataLoader(dataset, batch_size=batch_size, shuffle=shuffle)

# === 2. KONFIGURACE CURRICULA ===
DATA_DIR = './generated_data_synthetic_controlled'

# Inicializace manažera (teď je to jen wrapper pro načítání souborů)
data_manager = NavigationDataManager(DATA_DIR)

# Definice fází (zde řídíš, jak se trénink vyvíjí)
curriculum_schedule = [
    # FÁZE 1: Warm-up (Krátké sekvence)
    {
        'phase_id': 1,
        'seq_len': 10,
        'epochs': 500,
        'lr': 1e-3,
        'batch_size': 256
    },
```

```
    # FÁZE 2: Stabilizace (Střední délka)
    {
        'phase_id': 2,
        'seq_len': 100,
        'epochs': 200,
        'lr': 1e-4,
        'batch_size': 256
    },

    # FÁZE 3: Long-term Reality (Plná délka)
    {
        'phase_id': 3,
        'seq_len': 300,
        'epochs': 200,
        'lr': 1e-5,
        'batch_size': 128       # Menší batch kvůli paměti GPU u dlouhých␣
 ↪sekvencí
    }
]

# === 3. NAČÍTÁNÍ DO PAMĚTI (CACHING) ===
print("\n=== NAČÍTÁNÍ RAW DAT Z DISKU (BEZ EXT. NORMALIZACE) ===")
datasets_cache = {}

for phase in curriculum_schedule:
    seq_len = phase['seq_len']
    bs = phase['batch_size']

    print(f"  Načítám Fázi {phase['phase_id']}: Seq={seq_len} | Batch={bs} ...")

    try:
        # Použití DataManageru
        train_loader = data_manager.get_dataloader(seq_len=seq_len,␣
 ↪split='train', shuffle=True, batch_size=bs)
        val_loader = data_manager.get_dataloader(seq_len=seq_len, split='val',␣
 ↪shuffle=False, batch_size=bs)

        # Uložení do cache
        datasets_cache[phase['phase_id']] = (train_loader, val_loader)

        # Rychlá kontrola pro jistotu
        x_ex, y_ex = next(iter(train_loader))
        if phase['phase_id'] == 1:
            print(f"     Ukázka RAW dat (y): {y_ex[0, 0, :].tolist()}")
            # Měl bys vidět velká čísla (např. 250.0) a malá (0.2), ne ~0.0
```

```
    except FileNotFoundError as e:
        print(f"    CHYBA: {e}")
        # raise e # Odkomentuj, pokud chceš, aby to spadlo při chybě

print("\n Data připravena. Normalizaci řeší model.")
```

```
=== NAČÍTÁNÍ RAW DAT Z DISKU (BEZ EXT. NORMALIZACE) ===
 Načítám Fázi 1: Seq=10 | Batch=256 …
    Ukázka RAW dat (y): [323.7707824707031, -13.519903182983398,
-29.721908569335938]
 Načítám Fázi 2: Seq=100 | Batch=256 …
 Načítám Fázi 3: Seq=300 | Batch=128 …

 Data připravena. Normalizaci řeší model.
```

[7]:
```
import torch
import torch.nn.functional as F
import numpy as np
from copy import deepcopy

def gaussian_nll_safe(target, preds, var, min_var=1e-6, max_error_sq=100.0):
    """
    Bezpečná NLL loss funkce.
    """
    safe_var = var + min_var
    error_sq = (preds - target) ** 2
    # Clampujeme velikost chyby v čitateli, aby loss neexplodovala,
    # ale gradient do variance (ve jmenovateli) zůstal zachován.
    error_sq_clamped = torch.clamp(error_sq, max=max_error_sq)
    nll = 0.5 * (torch.log(safe_var) + error_sq_clamped / safe_var)
    return nll.mean()

def train_BayesianKalmanNet_TwoPhase(
    model, train_loader, val_loader, device,
    total_train_iter, learning_rate, clip_grad,
    J_samples, validation_period, logging_period,
    mse_warmup_iters=0,  # <--- NOVÝ PARAMETR: Kolik iterací trénovat jen na MSE
    weight_decay_=1e-5
):
    # Optimizer
    optimizer = torch.optim.AdamW(model.parameters(), lr=learning_rate,␣
 ↪weight_decay=weight_decay_)

    # Scheduler (volitelný, zde vypnutý pro jednoduchost)
    # scheduler = torch.optim.lr_scheduler.ReduceLROnPlateau(optimizer,␣
 ↪mode='min', factor=0.5, patience=50)
```

```
    best_val_metric = float('inf') # Buď MSE nebo ANEES podle fáze
    best_model_state = None
    best_iter_count = 0
    train_iter_count = 0
    done = False

    print(f"  START Two-Phase Training")
    print(f"    Phase 1: MSE Warmup (0 - {mse_warmup_iters} iters)")
    print(f"    Phase 2: NLL Optimization ({mse_warmup_iters} -␣
↪{total_train_iter} iters)")

    while not done:
        model.train()
        for x_true_batch, y_meas_batch in train_loader:
            if train_iter_count >= total_train_iter: done = True; break

            # Detekce NaN v datech
            if torch.isnan(x_true_batch).any():
                print(f"!!! SKIP BATCH iter {train_iter_count}: NaN found in␣
↪x_true !!!")
                continue

            x_true_batch = x_true_batch.to(device)
            y_meas_batch = y_meas_batch.to(device)
            batch_size, seq_len, _ = x_true_batch.shape

            # --- Training Step ---
            optimizer.zero_grad()

            all_trajectories_for_ensemble = []
            all_regs_for_ensemble = []

            # 1. Ensemble Forward Pass
            for j in range(J_samples):
                model.reset(batch_size=batch_size, initial_state=x_true_batch[:
↪, 0, :])
                current_trajectory_x_hats = []
                current_trajectory_regs = []

                for t in range(1, seq_len):
                    y_t = y_meas_batch[:, t, :]
                    x_filtered_t, reg_t = model.step(y_t)

                    if torch.isnan(x_filtered_t).any():
                        raise ValueError(f"NaN in x_filtered_t at sample {j},␣
↪step {t}")
```

```
                    current_trajectory_x_hats.append(x_filtered_t)
                    current_trajectory_regs.append(reg_t)

                all_trajectories_for_ensemble.append(torch.
↪stack(current_trajectory_x_hats, dim=1))
                all_regs_for_ensemble.append(torch.sum(torch.
↪stack(current_trajectory_regs)))

            # 2. Statistiky Ensemble
            ensemble_trajectories = torch.stack(all_trajectories_for_ensemble,␣
↪dim=0)
            x_hat_sequence = ensemble_trajectories.mean(dim=0)

            # Epistemická variance
            cov_diag_sequence = ensemble_trajectories.var(dim=0) + 1e-9

            # Normalizovaná regularizace (na délku sekvence)
            regularization_loss = torch.stack(all_regs_for_ensemble).mean() /␣
↪seq_len

            target_sequence = x_true_batch[:, 1:, :]

            # --- 3. VÝPOČET LOSS (Dvě fáze) ---

            # Vždy spočítáme obojí pro logování
            mse_loss = F.mse_loss(x_hat_sequence, target_sequence)
            nll_loss = gaussian_nll_safe(
                target=target_sequence,
                preds=x_hat_sequence,
                var=cov_diag_sequence,
                min_var=1e-5,
                max_error_sq=100.0
            )

            # Rozhodování o optimalizační Loss
            loss_mode = ""
            if train_iter_count < mse_warmup_iters:
                # FÁZE 1: Warmup na MSE
                # Ignorujeme NLL, soustředíme se na trefení trajektorie
                loss = mse_loss + regularization_loss
                loss_mode = "MSE_WARMUP"
            else:
                # FÁZE 2: NLL Optimalizace
                # Minimalizujeme NLL (kalibrace nejistoty + přesnost)
                # MSE zde není přímo v gradientu (je schované v NLL), ale␣
↪logujeme ho
```

```
                loss = nll_loss + regularization_loss
                loss_mode = "NLL_OPTIM"

            if torch.isnan(loss):
                print("Collapse detected (NaN loss)"); done = True; break

            loss.backward()

            # --- DIAGNOSTIKA GRADIENTŮ ---
            if train_iter_count % logging_period == 0:
                total_norm = 0.0
                max_grad = 0.0
                nan_grad_detected = False
                for p in model.parameters():
                    if p.grad is not None:
                        param_norm = p.grad.data.norm(2).item()
                        total_norm += param_norm ** 2
                        p_max = p.grad.data.abs().max().item()
                        if p_max > max_grad: max_grad = p_max
                        if torch.isnan(p.grad).any():
                            nan_grad_detected = True
                total_norm = total_norm ** 0.5

                if nan_grad_detected:
                     print(f"!!! WARNING: NaN gradient detected at iter␣
↪{train_iter_count} !!!")

            if clip_grad > 0:
                torch.nn.utils.clip_grad_norm_(model.parameters(), clip_grad)

            optimizer.step()
            train_iter_count += 1

            # --- LOGGING ---
            if train_iter_count % logging_period == 0:
                with torch.no_grad():
                    # Statistiky variance
                    min_variance = cov_diag_sequence.min().item()
                    max_variance = cov_diag_sequence.max().item()
                    mean_variance = cov_diag_sequence.mean().item()

                    # Dropout pravděpodobnosti
                    p1 = torch.sigmoid(model.dnn.concrete_dropout1.p_logit).
↪item()
                    p2 = torch.sigmoid(model.dnn.concrete_dropout2.p_logit).
↪item()
```

```
                    # Chyba v metrech (L1)
                    diff = x_hat_sequence - target_sequence
                    mean_error = diff.abs().mean().item()

                print(f"--- Iter [{train_iter_count}/{total_train_iter}]␣
↪({loss_mode}) ---")
                print(f"    Total Loss:     {loss.item():.4f}")
                print(f"    MSE (Metric):   {mse_loss.item():.4f}")
                print(f"    NLL (Metric):   {nll_loss.item():.4f}")
                print(f"    Reg Loss:       {regularization_loss.item():.6f}")
                print(f"    Var Stats:      Min={min_variance:.2e},␣
↪Max={max_variance:.2e}, Mean={mean_variance:.2e}")
                print(f"    Mean Error L1:  {mean_error:.4f} m")
                print(f"    Grad Norm:      {total_norm:.4f} (Max: {max_grad:.
↪4f})")
                print(f"    Dropout:        p1={p1:.4f}, p2={p2:.4f}")

# --- VALIDATION (Upravená část) ---
            if train_iter_count > 0 and train_iter_count % validation_period ==␣
↪0:
                print(f"\n--- Validation at iteration {train_iter_count} ---")
                model.eval()
                val_mse_list = []

                # Pro ANEES
                all_val_x_true, all_val_x_hat, all_val_P_hat = [], [], []

                with torch.no_grad():
                    for x_true_val, y_meas_val in val_loader:
                        v_bs, v_seq, _ = x_true_val.shape
                        x_true_val = x_true_val.to(device)
                        y_meas_val = y_meas_val.to(device)

                        val_ensemble_trajs = []

                        for j in range(J_samples):
                            model.reset(batch_size=v_bs,␣
↪initial_state=x_true_val[:, 0, :])
                            v_x_hats = []
                            for t in range(1, v_seq):
                                est, _ = model.step(y_meas_val[:, t, :])
                                v_x_hats.append(est)
                            val_ensemble_trajs.append(torch.stack(v_x_hats,␣
↪dim=1))
```

```
                        val_ens_stack = torch.stack(val_ensemble_trajs, dim=0)␣
↪# [J, B, T, D]
                        val_mean = val_ens_stack.mean(dim=0)
                        val_var_diag = val_ens_stack.var(dim=0) + 1e-9

                        val_mse_list.append(F.mse_loss(val_mean, x_true_val[:,␣
↪1:, :]).item())

                        # --- Příprava dat pro ANEES ---
                        # 1. Stavy (Ground Truth a Odhad)
                        # Přidáme startovní bod (t=0), abychom měli celou␣
↪sekvenci
                        full_x_hat = torch.cat([x_true_val[:, 0, :].
↪unsqueeze(1), val_mean], dim=1)

                        # 2. Kovariance (P)
                        # Vytvoříme plné matice 4x4 z diagonálního rozptylu
                        val_covs_full = torch.zeros(v_bs, v_seq-1, 4, 4,␣
↪device=device)
                        # Rychlá vektorizovaná konstrukce diagonály
                        # (Místo cyklu přes batch a čas použijeme diag_embed,␣
↪pokud to PyTorch verze umí)
                        try:
                            val_covs_full = torch.diag_embed(val_var_diag)
                        except:
                            # Fallback pro starší verze nebo pokud to selže
                            for b in range(v_bs):
                                for t in range(v_seq-1):
                                    val_covs_full[b, t] = torch.
↪diag(val_var_diag[b, t])

                        # P0 (počáteční nejistota - malá)
                        P0 = torch.eye(4, device=device).unsqueeze(0).
↪unsqueeze(0).repeat(v_bs, 1, 1, 1) * 1e-6
                        full_P_hat = torch.cat([P0, val_covs_full], dim=1)

                        all_val_x_true.append(x_true_val.cpu())
                        all_val_x_hat.append(full_x_hat.cpu())
                        all_val_P_hat.append(full_P_hat.cpu())

                avg_val_mse = np.mean(val_mse_list)

                # --- VÝPOČET ANEES ---
                try:
                    cat_true = torch.cat(all_val_x_true, dim=0)
                    cat_hat = torch.cat(all_val_x_hat, dim=0)
```

```
                    cat_P = torch.cat(all_val_P_hat, dim=0)

                    # Voláme vaši funkci (předpokládám, že je v modulu␣
↪'trainer' nebo 'utils')
                    # Pokud ji nemáte naimportovanou, musíte ji definovat.
                    # Zde používám 'trainer.calculate_anees_vectorized' z␣
↪vašeho původního kódu
                    if hasattr(trainer, 'calculate_anees_vectorized'):
                        avg_val_anees = trainer.
↪calculate_anees_vectorized(cat_true, cat_hat, cat_P)
                    else:
                        avg_val_anees = float('nan') # Placeholder

                except Exception as e:
                    print(f"Error calculating ANEES: {e}")
                    avg_val_anees = float('nan')

                print(f"  Avg MSE: {avg_val_mse:.4f} | Avg ANEES:␣
↪{avg_val_anees:.4f}")

                # --- LOGIKA UKLÁDÁNÍ (Smart Saving) ---
                # Ukládáme, pokud se zlepší MSE (priorita 1).
                # Volitelně: Ve fázi NLL by se dalo ukládat, pokud se zlepší␣
↪ANEES (k ideální 4.0),
                # ale to je riskantní, pokud by MSE vyletělo.
                # Zůstaňme u MSE jako "kotvy kvality".

                current_metric = avg_val_mse
                if current_metric < best_val_metric:
                    print(f"  >>> New Best Model! (MSE: {best_val_metric:.4f}␣
↪-> {current_metric:.4f}) <<<")
                    best_val_metric = current_metric
                    best_iter_count = train_iter_count
                    best_model_state = deepcopy(model.state_dict())

                print("-" * 50)
                model.train()

    print("\nTraining completed.")
    if best_model_state:
        print(f"Loading best model from iteration {best_iter_count}")
        model.load_state_dict(best_model_state)

    return {"final_model": model}
```

[8]:
```python
import torch
import torch.nn.functional as F
import numpy as np
from copy import deepcopy
def gaussian_nll_safe(target, preds, var, min_var=1e-6, max_error_sq=100.0):
    # 1. Bezpečná variance (epsilon) - správně
    safe_var = var + min_var

    # 2. Kvadratická chyba
    error_sq = (preds - target) ** 2

    # 3. === OPRAVA ===
    # Clampujeme "velikost chyby", nikoliv "velikost trestu".
    # Tím říkáme: "Pokud je chyba větší než 10m (100m^2), chovej se, jako by byla 10m."
    # Vzorec zůstává: Const / var.
    # Derivace je: -Const / var^2. (To je záporné číslo -> zvyšování var snižuje Loss -> SPRÁVNĚ!)
    error_sq_clamped = torch.clamp(error_sq, max=max_error_sq)

    # 4. Výpočet s oříznutou chybou
    nll = 0.5 * (torch.log(safe_var) + error_sq_clamped / safe_var)

    return nll.mean()

def train_BayesianKalmanNet_Hybrid(
    model, train_loader, val_loader, device,
    total_train_iter, learning_rate, clip_grad,
    J_samples, validation_period, logging_period,
    warmup_iterations=0, weight_decay_=1e-5,
    lambda_mse=100.0  # <--- NOVÝ PARAMETR: Kotva pro MSE
):
    # torch.autograd.set_detect_anomaly(True)
    optimizer = torch.optim.AdamW(model.parameters(), lr=learning_rate, weight_decay=weight_decay_)

    # Scheduler: Pokud se loss zasekne, snížíme LR (pomáhá stabilizovat konvergenci)
    # scheduler = torch.optim.lr_scheduler.ReduceLROnPlateau(
    #     optimizer, mode='min', factor=0.5, patience=50
    # )

    best_val_anees = float('inf')
    score_at_best = {"val_nll": 0.0, "val_mse": 0.0}
    best_iter_count = 0
    best_model_state = None
    train_iter_count = 0
```

```
    done = False

    print(f"  START Hybrid Training: Loss = NLL + {lambda_mse} * MSE")
    print(f"    Logging period: {logging_period} iterations")

    while not done:
        model.train()
        for x_true_batch, y_meas_batch in train_loader:
            if train_iter_count >= total_train_iter: done = True; break
            if torch.isnan(x_true_batch).any():
                print(f"!!! SKIP BATCH iter {train_iter_count}: NaN found in␣
↪x_true (Ground Truth) !!!")
                continue

            x_true_batch = x_true_batch.to(device)
            y_meas_batch = y_meas_batch.to(device)

            # --- Training ---
            optimizer.zero_grad()
            batch_size, seq_len, _ = x_true_batch.shape

            all_trajectories_for_ensemble = []
            all_regs_for_ensemble = []

            # 1. Ensemble Forward Pass
            for j in range(J_samples):
                model.reset(batch_size=batch_size, initial_state=x_true_batch[:
↪, 0, :])
                current_trajectory_x_hats = []
                current_trajectory_regs = []
                for t in range(1, seq_len):
                    y_t = y_meas_batch[:, t, :]
                    x_filtered_t, reg_t = model.step(y_t)
                    if torch.isnan(x_filtered_t).any():
                            raise ValueError(f"NaN in x_filtered_t at sample␣
↪{j}, step {t}")
                    current_trajectory_x_hats.append(x_filtered_t)
                    current_trajectory_regs.append(reg_t)
                all_trajectories_for_ensemble.append(torch.
↪stack(current_trajectory_x_hats, dim=1))
                all_regs_for_ensemble.append(torch.sum(torch.
↪stack(current_trajectory_regs)))

            # 2. Statistiky Ensemble
            ensemble_trajectories = torch.stack(all_trajectories_for_ensemble,␣
↪dim=0)
```

```
            x_hat_sequence = ensemble_trajectories.mean(dim=0)

            # Epistemická variance (čistý rozptyl sítě)
            # Přičítáme 1e-9 jen proti dělení nulou, není to "noise floor"
            cov_diag_sequence = ensemble_trajectories.var(dim=0) + 1e-9

            regularization_loss = torch.stack(all_regs_for_ensemble).mean()/
↪seq_len
            target_sequence = x_true_batch[:, 1:, :]

            # --- 3. VÝPOČET HYBRIDNÍ LOSS ---

            # A) MSE Část (Přesnost)
            mse_loss = F.mse_loss(x_hat_sequence, target_sequence)

            # B) NLL Část (Konzistence)
            # 0.5 * (log(var) + (target - pred)^2 / var)
            cov_diag_clamped = torch.clamp(cov_diag_sequence, min=1e-4, max=1e6)
            error_sq = (x_hat_sequence - target_sequence) ** 2
            nll_term = 0.5 * (torch.log(cov_diag_clamped) + error_sq /␣
↪cov_diag_clamped)
            nll_loss = gaussian_nll_safe(
                target=target_sequence,
                preds=x_hat_sequence,
                var=cov_diag_sequence,
                min_var=1e-5,       # Epsilon pro stabilitu
                max_error_sq=100.0 # Ořezání extrémních chyb (pokud je␣
↪implementováno)
            )
            mean_var = cov_diag_sequence.mean()
            var_penalty = torch.relu(mean_var - 100.0) * 0.01

            # C) Celková Loss (Hybrid)
            # Zde je ta magie: I když NLL chce utéct s variancí, lambda_mse *␣
↪mse ho drží zpátky
            weighted_mse = lambda_mse * mse_loss
            loss = nll_loss + weighted_mse + regularization_loss * 10.0 +␣
↪var_penalty

            if torch.isnan(loss):
                print("Collapse detected (NaN loss)"); done = True; break

            loss.backward()

            # --- DIAGNOSTIC LOGGING (Gradients) ---
            # Zaznamenáme statistiky gradientů před oříznutím (clippingem)
            if train_iter_count % logging_period == 0:
```

```
                total_norm = 0.0
                max_grad = 0.0
                min_grad = float('inf')
                nan_grad_detected = False
                for p in model.parameters():
                    if p.grad is not None:
                        param_norm = p.grad.data.norm(2).item()
                        total_norm += param_norm ** 2
                        p_max = p.grad.data.abs().max().item()
                        p_min = p.grad.data.abs().min().item()
                        if p_max > max_grad: max_grad = p_max
                        if p_min < min_grad: min_grad = p_min
                        if torch.isnan(p.grad).any():
                            nan_grad_detected = True
                total_norm = total_norm ** 0.5

                if nan_grad_detected:
                     print(f"!!! WARNING: NaN gradient detected at iter␣
↪{train_iter_count} !!!")

            if clip_grad > 0: torch.nn.utils.clip_grad_norm_(model.
↪parameters(), clip_grad)
            optimizer.step()
            train_iter_count += 1

            # --- Logging ---
            diff = x_hat_sequence - target_sequence
            mean_error = diff.abs().mean().item()
            min_variance = cov_diag_sequence.min().item()
            max_variance = cov_diag_sequence.max().item()
            mean_variance = cov_diag_sequence.mean().item()

            if train_iter_count % logging_period == 0:
                with torch.no_grad():
                    # Zjistíme dropout pravděpodobnosti (jen pro info)
                    p1 = torch.sigmoid(model.dnn.concrete_dropout1.p_logit).
↪item()
                    p2 = torch.sigmoid(model.dnn.concrete_dropout2.p_logit).
↪item()

                print(f"--- Iter [{train_iter_count}/{total_train_iter}] ---")
                print(f"    Total Loss:     {loss.item():.4f}")
                print(f"    MSE (Raw):      {mse_loss.item():.6f}")
                print(f"    MSE (Weighted): {weighted_mse.item():.4f}␣
↪(lambda={lambda_mse})")
                print(f"    NLL Component:  {nll_loss.item():.4f}")
                print(f"    Reg Loss:       {regularization_loss.item():.6f}")
```

```
                print(f"    Variance stats: Min={min_variance:.2e},␣
↪Max={max_variance:.2e}, Mean={mean_variance:.2e}")
                print(f"    Mean Error L1:  {mean_error:.4f}")
                print(f"    Grad Norm:      {total_norm:.4f} (Max abs grad:␣
↪{max_grad:.4f})")
                print(f"    Dropout probs:  p1={p1:.4f}, p2={p2:.4f}")

                # Check pro "Variance collapse"
                if mean_variance < 1e-8:
                    print("    !!! WARNING: Variance is extremely low (Collapse␣
↪risk) !!!")

            # --- Validation step ---
            if train_iter_count > 0 and train_iter_count % validation_period ==␣
↪0:
                # Step scheduleru podle trénovací loss (nebo validace, pokud␣
↪bys to předělal)
                # scheduler.step(loss)

                print(f"\n--- Validation at iteration {train_iter_count} ---")
                model.eval()
                val_mse_list = []
                all_val_x_true_cpu, all_val_x_hat_cpu, all_val_P_hat_cpu = [],␣
↪[], []

                with torch.no_grad():
                    for x_true_val_batch, y_meas_val_batch in val_loader:
                        val_batch_size, val_seq_len, _ = x_true_val_batch.shape
                        x_true_val_batch = x_true_val_batch.to(device)
                        y_meas_val_batch = y_meas_val_batch.to(device)
                        val_ensemble_trajectories = []
                        for j in range(J_samples):
                            model.reset(batch_size=val_batch_size,␣
↪initial_state=x_true_val_batch[:, 0, :])
                            val_current_x_hats = []
                            for t in range(1, val_seq_len):
                                y_t_val = y_meas_val_batch[:, t, :]
                                x_filtered_t, _ = model.step(y_t_val)
                                val_current_x_hats.append(x_filtered_t)
                            val_ensemble_trajectories.append(torch.
↪stack(val_current_x_hats, dim=1))

                        # Agregace validace
                        val_ensemble = torch.stack(val_ensemble_trajectories,␣
↪dim=0)
                        val_preds_seq = val_ensemble.mean(dim=0)
```

```
                    val_target_seq = x_true_val_batch[:, 1:, :]
                    val_mse_list.append(F.mse_loss(val_preds_seq,␣
↪val_target_seq).item())

                    # Příprava pro ANEES
                    initial_state_val = x_true_val_batch[:, 0, :].
↪unsqueeze(1)
                    full_x_hat = torch.cat([initial_state_val,␣
↪val_preds_seq], dim=1)

                    # Epistemická variance
                    val_covs_diag = val_ensemble.var(dim=0) + 1e-9

                    # Vytvoření diagonálních matic P
                    # (Zjednodušená konstrukce pro ANEES calc)
                    # Pro přesné ANEES bychom měli dělat outer product,
                    # ale diagonála z var() je dobrá aproximace pro BKN
                    val_covs_full = torch.zeros(val_batch_size,␣
↪val_seq_len-1, 4, 4, device=device)
                    for b in range(val_batch_size):
                        for t in range(val_seq_len-1):
                            val_covs_full[b, t] = torch.
↪diag(val_covs_diag[b, t])

                    P0 = model.system_model.P0.unsqueeze(0).
↪repeat(val_batch_size, 1, 1).unsqueeze(1)
                    full_P_hat = torch.cat([P0, val_covs_full], dim=1)

                    all_val_x_true_cpu.append(x_true_val_batch.cpu())
                    all_val_x_hat_cpu.append(full_x_hat.cpu())
                    all_val_P_hat_cpu.append(full_P_hat.cpu())

            avg_val_mse = np.mean(val_mse_list)
            final_x_true_list = torch.cat(all_val_x_true_cpu, dim=0)
            final_x_hat_list = torch.cat(all_val_x_hat_cpu, dim=0)
            final_P_hat_list = torch.cat(all_val_P_hat_cpu, dim=0)

            # Výpočet ANEES
            try:
                avg_val_anees = trainer.
↪calculate_anees_vectorized(final_x_true_list, final_x_hat_list,␣
↪final_P_hat_list)
            except Exception as e:
                print(f"  !!! Error calculating ANEES: {e}")
                avg_val_anees = float('nan')
```

```
                print(f"  Average MSE: {avg_val_mse:.4f}, Average ANEES:␣
↪{avg_val_anees:.4f}")

                # Ukládání modelu:
                if not np.isnan(avg_val_anees) and avg_val_anees <␣
↪best_val_anees and avg_val_anees > 0:
                    print(f"  >>> New best VALIDATION ANEES! Saving model. (Old:
↪ {best_val_anees:.4f} -> New: {avg_val_anees:.4f}) <<<")
                    best_val_anees = avg_val_anees
                    best_iter_count = train_iter_count
                    score_at_best['val_mse'] = avg_val_mse
                    best_model_state = deepcopy(model.state_dict())
                print("-" * 50)
                model.train()

    print("\nTraining completed.")
    if best_model_state:
        print(f"Loading best model from iteration {best_iter_count} with ANEES␣
↪{best_val_anees:.4f}")
        model.load_state_dict(best_model_state)
    else:
        print("No best model was saved; returning last state.")

    return {
        "best_val_anees": best_val_anees,
        "best_val_nll": score_at_best['val_nll'],
        "best_val_mse": score_at_best['val_mse'],
        "best_iter": best_iter_count,
        "final_model": model
    }
```

[9]:
```
import torch
import torch.nn as nn
import torch.nn.functional as F
import numpy as np
from copy import deepcopy

# Předpokládáme existenci helperu (pokud ne, definujte ho jako v minulé␣
↪odpovědi)
def gaussian_nll_safe(target, preds, var, min_var=1e-6, max_error_sq=100.0):
    safe_var = var + min_var
    error_sq = (preds - target) ** 2
    error_sq_clamped = torch.clamp(error_sq, max=max_error_sq)
    nll = 0.5 * (torch.log(safe_var) + error_sq_clamped / safe_var)
    return nll.mean()
```

```python
def train_BayesianKalmanNet_TBPTT_TwoPhase(
    model, train_loader, val_loader, device,
    total_train_iter, learning_rate, clip_grad=1.0,
    J_samples=5, tbptt_steps=20,
    validation_period=50, logging_period=10,
    mse_warmup_iters=0, # Počet updatů pro MSE fázi
    weight_decay_=1e-5
):
    optimizer = torch.optim.AdamW(model.parameters(), lr=learning_rate,␣
 ↪weight_decay=weight_decay_)

    best_val_mse = float('inf')
    best_model_state = None
    train_iter_count = 0 # Počítadlo updatů (gradient steps)
    done = False

    print(f" START TBPTT Two-Phase Training (Window={tbptt_steps})")
    print(f"    Phase 1: MSE Warmup (0 - {mse_warmup_iters} steps)")
    print(f"    Phase 2: NLL Optimization ({mse_warmup_iters} -␣
 ↪{total_train_iter} steps)")

    while not done:
        model.train()
        for x_true_batch, y_meas_batch in train_loader:
            if train_iter_count >= total_train_iter: done = True; break

            x_true_batch = x_true_batch.to(device)
            y_meas_batch = y_meas_batch.to(device)

            batch_size, seq_len, dim_x = x_true_batch.shape

            # --- SUPER BATCH (Vektorizace) ---
            x_true_super = x_true_batch.repeat_interleave(J_samples, dim=0)
            y_meas_super = y_meas_batch.repeat_interleave(J_samples, dim=0)
            super_batch_size = x_true_super.shape[0]

            # 1. Reset na začátku sekvence
            model.reset(batch_size=super_batch_size,␣
 ↪initial_state=x_true_super[:, 0, :])

            # Gradienty nulujeme před sekvencí
            optimizer.zero_grad()

            # 2. TBPTT Smyčka přes okna
            for t_start in range(1, seq_len, tbptt_steps):
                if train_iter_count >= total_train_iter: done = True; break
```

```
                t_end = min(t_start + tbptt_steps, seq_len)
                current_window_len = t_end - t_start
                if current_window_len <= 0: continue

                # A) Forward pass oknem
                window_x_preds = []
                window_regs = []

                for t in range(t_start, t_end):
                    y_t = y_meas_super[:, t, :]
                    x_est, reg = model.step(y_t)
                    window_x_preds.append(x_est)
                    window_regs.append(reg)

                # B) Zpracování výsledků okna
                preds_super = torch.stack(window_x_preds, dim=1) # [Batch*J,␣
↪Window, 4]
                regs_super = torch.stack(window_regs)

                # Reshape pro statistiku [Batch, J, Window, 4]
                preds_reshaped = preds_super.view(batch_size, J_samples,␣
↪current_window_len, dim_x)

                x_hat_seq = preds_reshaped.mean(dim=1)
                cov_diag_seq = preds_reshaped.var(dim=1) + 1e-9

                target_seq = x_true_batch[:, t_start:t_end, :]

                # C) Výpočet Loss (Two-Phase)
                mse_loss = F.mse_loss(x_hat_seq, target_seq)

                # NLL (použijeme bezpečnou funkci)
                nll_loss = gaussian_nll_safe(target_seq, x_hat_seq,␣
↪cov_diag_seq, max_error_sq=100.0)

                # Regularizace (průměr na krok)
                reg_loss = regs_super.mean()

                # Rozhodování o Loss
                if train_iter_count < mse_warmup_iters:
                    loss = mse_loss + reg_loss
                    mode = "MSE"
                else:
                    # NLL Fáze (můžeme přidat malou kotvu MSE, např. 0.1, pro␣
↪stabilitu)
                    loss = nll_loss + reg_loss + (0.1 * mse_loss)
                    mode = "NLL"
```

```
                # D) Backward & Update
                loss.backward()

                if clip_grad > 0:
                    torch.nn.utils.clip_grad_norm_(model.parameters(),␣
↪clip_grad)

                optimizer.step()
                optimizer.zero_grad() # Důležité: nulujeme po každém kroku TBPTT

                # E) Detach Hidden State (Klíčové pro TBPTT)
                model.detach_hidden()

                train_iter_count += 1

                # --- Logging ---
                if train_iter_count % logging_period == 0:
                    with torch.no_grad():
                        p1 = torch.sigmoid(model.dnn.concrete_dropout1.p_logit).
↪item()
                        p2 = torch.sigmoid(model.dnn.concrete_dropout2.p_logit).
↪item()
                        diff = x_hat_seq - target_seq
                        mae = diff.abs().mean().item()

                    print(f"Iter {train_iter_count} ({mode}): Loss {loss.item():
↪.4f} | MSE {mse_loss.item():.2f} | NLL {nll_loss.item():.2f} | MAE {mae:.
↪2f}m")
                    print(f"    Dropout: p1={p1:.3f}, p2={p2:.3f} | VarMean:␣
↪{cov_diag_seq.mean().item():.1f}")

                # --- Validation (Uvnitř TBPTT smyčky) ---
                if train_iter_count % validation_period == 0:
                    model.eval()
                    val_mse_list = []

                    # Pro ANEES sběr
                    all_val_x_true, all_val_x_hat, all_val_P = [], [], []

                    with torch.no_grad():
                        for x_v, y_v in val_loader:
                            x_v, y_v = x_v.to(device), y_v.to(device)
                            b_v, s_v, _ = x_v.shape

                            # Validace běží Open-Loop na celé sekvenci (bez␣
↪TBPTT)
```

```
                            x_v_sup = x_v.repeat_interleave(J_samples, dim=0)
                            y_v_sup = y_v.repeat_interleave(J_samples, dim=0)

                            model.reset(batch_size=b_v*J_samples,␣
↪initial_state=x_v_sup[:,0,:])
                            preds_list = []

                            for ti in range(1, s_v):
                                est, _ = model.step(y_v_sup[:, ti, :])
                                preds_list.append(est)

                            preds_stack = torch.stack(preds_list, dim=1).
↪view(b_v, J_samples, s_v-1, 4)
                            val_mean = preds_stack.mean(dim=1)
                            val_var = preds_stack.var(dim=1) + 1e-9

                            val_mse_list.append(F.mse_loss(val_mean, x_v[:, 1:,␣
↪:]).item())

                            # Data pro ANEES
                            full_hat = torch.cat([x_v[:,0,:].unsqueeze(1),␣
↪val_mean], dim=1)
                            # Diagonal P construction
                            val_P_full = torch.zeros(b_v, s_v-1, 4, 4,␣
↪device=device)
                            for b in range(b_v):
                                for t in range(s_v-1):
                                    val_P_full[b,t] = torch.diag(val_var[b,t])
                            P0 = torch.eye(4, device=device).unsqueeze(0).
↪unsqueeze(0).repeat(b_v, 1, 1, 1)*1e-6
                            full_P = torch.cat([P0, val_P_full], dim=1)

                            all_val_x_true.append(x_v.cpu())
                            all_val_x_hat.append(full_hat.cpu())
                            all_val_P.append(full_P.cpu())

                    avg_val_mse = np.mean(val_mse_list)

                    # Calc ANEES
                    try:
                        cat_true = torch.cat(all_val_x_true, dim=0)
                        cat_hat = torch.cat(all_val_x_hat, dim=0)
                        cat_P = torch.cat(all_val_P, dim=0)
                        # Pokud máte funkci importovanou
                        if 'trainer' in globals() and hasattr(trainer,␣
↪'calculate_anees_vectorized'):
```

```
                            avg_anees = trainer.
↪calculate_anees_vectorized(cat_true, cat_hat, cat_P)
                        else:
                            avg_anees = 0.0
                    except:
                        avg_anees = float('nan')

                    print(f"\n--- VALIDATION: MSE {avg_val_mse:.2f} | ANEES␣
↪{avg_anees:.2f} ---")

                    # Ukládání (podle MSE)
                    if avg_val_mse < best_val_mse:
                        print(f"  >>> New Best Model! (Old: {best_val_mse:.2f}␣
↪-> New: {avg_val_mse:.2f}) <<<")
                        best_val_mse = avg_val_mse
                        best_model_state = deepcopy(model.state_dict())
                    print("-" * 40)

                    model.train()

    print("TBPTT Training completed.")
    if best_model_state:
        model.load_state_dict(best_model_state)
    return {"final_model": model}
```

[10]:
```
import torch
import torch.nn as nn
import torch.nn.functional as F
import numpy as np
from copy import deepcopy

def gaussian_nll_safe(target, preds, var, min_var=1e-6, max_error_sq=100.0):
    safe_var = var + min_var
    error_sq = (preds - target) ** 2
    error_sq_clamped = torch.clamp(error_sq, max=max_error_sq)
    nll = 0.5 * (torch.log(safe_var) + error_sq_clamped / safe_var)
    return nll.mean()

def train_BayesianKalmanNet_TBPTT_Windowed(
    model, train_loader, val_loader, device,
    total_train_iter, learning_rate, clip_grad=1.0,
    J_samples=5,
    tbptt_k=5,
    tbptt_w=20,
    validation_period=50, logging_period=10,
    mse_warmup_iters=0,
    weight_decay_=1e-5,
```

```
    lambda_mse=100.0
):
    optimizer = torch.optim.AdamW(model.parameters(), lr=learning_rate,␣
 ↪weight_decay=weight_decay_)

    best_val_mse = float('inf')
    best_model_state = None
    train_iter_count = 0
    done = False

    if not hasattr(model, 'detach_hidden'):
         raise AttributeError("Modelu chybí metoda 'detach_hidden()'.")

    print(f"  START TBPTT Windowed Training (k={tbptt_k}, w={tbptt_w})")
    print(f"    Phase 1: MSE Warmup (0 - {mse_warmup_iters} updates)")
    print(f"    Phase 2: NLL Optimization (> {mse_warmup_iters} updates)")

    while not done:
        model.train()
        for x_true_batch, y_meas_batch in train_loader:
            if train_iter_count >= total_train_iter: done = True; break

            x_true_batch = x_true_batch.to(device)
            y_meas_batch = y_meas_batch.to(device)

            batch_size, seq_len, dim_x = x_true_batch.shape

            x_true_super = x_true_batch.repeat_interleave(J_samples, dim=0)
            y_meas_super = y_meas_batch.repeat_interleave(J_samples, dim=0)
            super_batch_size = x_true_super.shape[0]

            # 1. Hard Reset na začátku sekvence
            # Vyčistíme vše, aby reset vytvořil správné velikosti
            if hasattr(model, 'h_prev'): model.h_prev = None
            if hasattr(model, 'x_filtered_t_minus_1'): model.
 ↪x_filtered_t_minus_1 = None
            if hasattr(model, 'x_filtered_t_minus_2'): model.
 ↪x_filtered_t_minus_2 = None

            model.reset(batch_size=super_batch_size,␣
 ↪initial_state=x_true_super[:, 0, :])

            # 2. TBPTT Loop
            for t_start in range(1, seq_len, tbptt_w):
                if train_iter_count >= total_train_iter: done = True; break

                t_end = min(t_start + tbptt_w, seq_len)
```

```
                current_window_len = t_end - t_start
                if current_window_len <= 0: continue

                optimizer.zero_grad()
                window_x_preds = []
                window_regs = []

                # A) Forward pass
                for t in range(t_start, t_end):
                    y_t = y_meas_super[:, t, :]
                    x_est, reg = model.step(y_t)
                    window_x_preds.append(x_est)
                    window_regs.append(reg)

                    if (t - t_start + 1) % tbptt_k == 0:
                        model.detach_hidden()

                model.detach_hidden()

                # B) Loss Calculation
                preds_super = torch.stack(window_x_preds, dim=1)
                regs_super = torch.stack(window_regs)
                preds_reshaped = preds_super.view(batch_size, J_samples,␣
↪current_window_len, dim_x)

                x_hat_seq = preds_reshaped.mean(dim=1)
                cov_diag_seq = preds_reshaped.var(dim=1) + 1e-9
                target_seq = x_true_batch[:, t_start:t_end, :]

                mse_loss = F.mse_loss(x_hat_seq, target_seq)
                nll_loss = gaussian_nll_safe(target_seq, x_hat_seq,␣
↪cov_diag_seq, max_error_sq=100.0)
                reg_loss = regs_super.mean()

                if train_iter_count < mse_warmup_iters:
                    loss = mse_loss + reg_loss
                    mode = "MSE_WARMUP"
                else:
                    loss = nll_loss + reg_loss + (lambda_mse * mse_loss)
                    mode = "NLL_OPTIM"

                # C) Update
                loss.backward()
                if clip_grad > 0:
                    torch.nn.utils.clip_grad_norm_(model.parameters(),␣
↪clip_grad)
                optimizer.step()
```

```
                optimizer.zero_grad()
                model.detach_hidden()

                train_iter_count += 1

                if train_iter_count % logging_period == 0:
                    with torch.no_grad():
                        diff = x_hat_seq - target_seq
                        mae = diff.abs().mean().item()
                    print(f"Iter {train_iter_count} ({mode}): Loss {loss.item():
↪.4f} | MSE {mse_loss.item():.2f} | NLL {nll_loss.item():.2f} | MAE {mae:.
↪2f}m")

                # --- VALIDATION WITH FULL STATE RESTORE (FIXED) ---
                if train_iter_count % validation_period == 0:
                    # 1. ULOŽIT KOMPLETNÍ STAV TRÉNINKU
                    # Musíme uložit úplně všechno, co se mění v čase t
                    train_state = {}

                    if model.h_prev is not None:
                        train_state['h_prev'] = model.h_prev.detach().clone()

                    train_state['x_filt_1'] = model.x_filtered_t_minus_1.
↪detach().clone()
                    train_state['x_pred_1'] = model.x_pred_t_minus_1.detach().
↪clone()

                    # !!! ZDE BYLA CHYBA: Musíme uložit i t-2 a y_t-1 !!!
                    if hasattr(model, 'x_filtered_t_minus_2') and model.
↪x_filtered_t_minus_2 is not None:
                        train_state['x_filt_2'] = model.x_filtered_t_minus_2.
↪detach().clone()

                    if hasattr(model, 'y_t_minus_1') and model.y_t_minus_1 is␣
↪not None:
                        train_state['y_1'] = model.y_t_minus_1.detach().clone()

                    if hasattr(model, 'P_t_minus_1') and model.P_t_minus_1 is␣
↪not None:
                        train_state['P'] = model.P_t_minus_1.detach().clone()

                    # 2. Spustit Validaci
                    model.eval()
                    val_mse_list = []
                    all_val_x_true, all_val_x_hat, all_val_P = [], [], []
```

```
                    with torch.no_grad():
                        for x_v, y_v in val_loader:
                            x_v, y_v = x_v.to(device), y_v.to(device)
                            b_v, s_v, _ = x_v.shape
                            x_v_sup = x_v.repeat_interleave(J_samples, dim=0)
                            y_v_sup = y_v.repeat_interleave(J_samples, dim=0)

                            # Hard reset pro validaci (smaže interní proměnné)
                            if hasattr(model, 'h_prev'): model.h_prev = None
                            model.reset(batch_size=b_v*J_samples,␣
↪initial_state=x_v_sup[:,0,:])

                            preds_list = []
                            for ti in range(1, s_v):
                                est, _ = model.step(y_v_sup[:, ti, :])
                                preds_list.append(est)

                            preds_stack = torch.stack(preds_list, dim=1).
↪view(b_v, J_samples, s_v-1, 4)
                            val_mean = preds_stack.mean(dim=1)
                            val_var = preds_stack.var(dim=1) + 1e-9
                            val_mse_list.append(F.mse_loss(val_mean, x_v[:, 1:,␣
↪:]).item())

                            # ANEES data
                            all_val_x_true.append(x_v.cpu())
                            all_val_x_hat.append(val_mean.cpu())

                            # Diagonální P pro ANEES
                            val_P_full = torch.zeros(b_v, s_v-1, 4, 4,␣
↪device=device)
                            for i in range(4): val_P_full[:, :, i, i] =␣
↪val_var[:, :, i]
                            P0 = torch.eye(4, device=device).unsqueeze(0).
↪unsqueeze(0).repeat(b_v, 1, 1, 1)*1e-6
                            full_P = torch.cat([P0, val_P_full], dim=1)
                            all_val_P.append(full_P.cpu())

                    avg_val_mse = np.mean(val_mse_list)

                    try:
                        cat_true = torch.cat(all_val_x_true, dim=0)
                        cat_hat = torch.cat(all_val_x_hat, dim=0)
                        cat_P = torch.cat(all_val_P, dim=0)
                        if 'trainer' in globals() and hasattr(trainer,␣
↪'calculate_anees_vectorized'):
```

```
                            avg_anees = trainer.
↪calculate_anees_vectorized(cat_true, cat_hat, cat_P)
                        else:
                            avg_anees = 0.0
                    except:
                        avg_anees = float('nan')

                    print(f"\n--- VALIDATION: MSE {avg_val_mse:.2f} | ANEES␣
↪{avg_anees:.2f} ---")

                    if avg_val_mse < best_val_mse:
                        best_val_mse = avg_val_mse
                        best_model_state = deepcopy(model.state_dict())
                        print(f"  >>> New Best Model! (MSE: {best_val_mse:.2f})␣
↪<<<")
                    print("-" * 40)

                    # 3. OBNOVIT KOMPLETNÍ STAV TRÉNINKU
                    model.train()
                    if 'h_prev' in train_state: model.h_prev =␣
↪train_state['h_prev']
                    model.x_filtered_t_minus_1 = train_state['x_filt_1']
                    model.x_pred_t_minus_1 = train_state['x_pred_1']

                    if 'x_filt_2' in train_state: model.x_filtered_t_minus_2 =␣
↪train_state['x_filt_2']
                    if 'y_1' in train_state: model.y_t_minus_1 =␣
↪train_state['y_1']
                    if 'P' in train_state: model.P_t_minus_1 = train_state['P']

    print("Training completed.")
    if best_model_state:
        model.load_state_dict(best_model_state)
    return {"final_model": model}
```

[11]:
```
import torch
import copy

# Předpokládám, že funkce train_BayesianKalmanNet_TBPTT je definována (z␣
↪předchozího kroku)
# from utils import train_BayesianKalmanNet_TBPTT  <-- pokud ji máte v souboru

# --- 1. DEFINICE CURRICULA ---
curriculum_schedule = [
    # FÁZE 1: Stabilizace (Krátké sekvence - stačí standardní trénink)
    {
```

```
        'phase_id': 1,
        'seq_len': 10,
        'iters': 700,
        'lr': 1e-3,
        'lambda_mse': 1.0,
        'clip_grad': 1.0,
        'use_tbptt': False,   # Tady TBPTT nepotřebujeme
        'mse_warmup_iters': 700 # Prvních 300 iterací se zaměříme pouze na MSE, pak přidáme NLL
    },
    {
        'phase_id': 2,
        'seq_len': 100,
        'iters': 2500,
        'lambda_mse': 0.5,
        'lr': 5e-5,
        'clip_grad': 1.0,
        'use_tbptt': True,    # <--- True
        'tbptt_w': 20,    # Delší okno pro lepší kontext
        'tbptt_k': 5,         # Hlubší gradient
        'mse_warmup_iters': 1250 # Prvních 1000 iterací se zaměříme pouze na MSE, pak přidáme NLL
    },
    {
        'phase_id': 3,
        'seq_len': 300,
        'iters': 2000,        # Více iterací, protože TBPTT dělá menší kroky
        'lr': 1e-6,           # Opatrné učení
        'clip_grad': 0.5,
        'lambda_mse': 0.1,
        'use_tbptt': True,    # <--- ZAPNUTO
        'tbptt_w': 20,    # Delší okno pro lepší kontext
        'tbptt_k': 5,         # Hlubší gradient
        'mse_warmup_iters': 500 # Prvních 1000 iterací se zaměříme pouze na MSE, pak přidáme NLL

    }
]

# --- 2. INICIALIZACE MODELU ---
print("=== INICIALIZACE BKN MODELU ===")
# Používáme váš osvědčený setup s vyšším hidden size
state_knet2 = TAN.StateBayesianKalmanNetTAN(
        system_model=system_model,
        device=device,
        hidden_size_multiplier=12,
        output_layer_multiplier=4,
```

```
        num_gru_layers=1,
        init_max_dropout=0.2, # Zdravý dropout pro exploration
        init_min_dropout=0.1
).to(device)

# --- 3. CURRICULUM LOOP ---
for phase in curriculum_schedule:
    phase_id = phase['phase_id']
    seq_len = phase['seq_len']

    # Kontrola dostupnosti dat
    if phase_id not in datasets_cache:
        print(f"  Skipping Phase {phase_id}: Data not in cache.")
        continue

    print(f"\n" + "="*60)
    print(f"  START PHASE {phase_id}: SeqLen {seq_len} | LR {phase['lr']} |␣
 ↪TBPTT: {phase['use_tbptt']}")
    print("="*60)

    train_loader_phase = datasets_cache[phase_id][0]
    val_loader_phase = datasets_cache[phase_id][1]

    # Rozhodování podle typu tréninku
    if phase['use_tbptt']:
        # === A) TBPTT Trénink (pro dlouhé sekvence) ===

        # Zkontrolujeme, zda model má metodu detach_hidden (prevence pádu)
        if not hasattr(state_knet2, 'detach_hidden'):
            raise AttributeError("Modelu chybí metoda 'detach_hidden()'!␣
 ↪Přidejte ji do třídy StateBayesianKalmanNetTAN.")

        result = train_BayesianKalmanNet_TBPTT_Windowed(
            model=state_knet2,
            train_loader=train_loader_phase,
            val_loader=val_loader_phase,
            device=device,
            total_train_iter=phase['iters'],
            learning_rate=phase['lr'],
            clip_grad=phase['clip_grad'],
            J_samples=10,               # V TBPTT můžeme mít menší J, protože␣
 ↪se průměruje častěji
            tbptt_w=phase.get('tbptt_w', 10), # Default 20 pokud není v configu
            tbptt_k=phase.get('tbptt_k', 2),
            validation_period=20,       # Validace každých 50 oken
            logging_period=10,
            mse_warmup_iters=phase['mse_warmup_iters']
```

```
        )

    else:
        # === B) Standardní Hybrid Trénink (pro krátké sekvence) ===
        print(f"   -> Using Standard Hybrid Training")
        result = train_BayesianKalmanNet_TwoPhase(
            model=state_knet2,
            train_loader=train_loader_phase,
            val_loader=val_loader_phase,
            device=device,
            total_train_iter=phase['iters'],
            learning_rate=phase['lr'],
            clip_grad=phase['clip_grad'],
            J_samples=5,
            validation_period=10,
            logging_period=10,
            mse_warmup_iters=phase['mse_warmup_iters'],
            weight_decay_=1e-3
        )

    # Uložení a kontrola
    save_path = f"bkn_curriculum_phase{phase_id}_len{seq_len}.pth"
    torch.save(state_knet2.state_dict(), save_path)
    print(f"  Fáze {phase_id} dokončena. Model uložen do: {save_path}")

    # Jednoduchá kontrola proti divergenci (pokud best_val neexistuje nebo je␣
↪inf)
    # Pozn: TBPTT funkce vrací slovník, hybrid taky, ujistíme se, že klíče sedí
    # TBPTT momentálně vrací jen {'final_model': model}, pro pokročilou␣
↪kontrolu by to chtělo vrátit i loss.
    # Ale pro teď to necháme běžet dál.

print("\n  Celý trénink dokončen.")
```

```
=== INICIALIZACE BKN MODELU ===
INFO: Aplikuji upravenou inicializaci pro BKN.
DEBUG: Výstupní vrstva inicializována konzervativně (interval -0.1 až 0.1).

============================================================
  START PHASE 1: SeqLen 10 | LR 0.001 | TBPTT: False
============================================================
   -> Using Standard Hybrid Training
  START Two-Phase Training
    Phase 1: MSE Warmup (0 - 700 iters)
    Phase 2: NLL Optimization (700 - 700 iters)
--- Iter [10/700] (MSE_WARMUP) ---
    Total Loss:     53.1780
```

```
    MSE (Metric):   53.1765
    NLL (Metric):   21.3128
    Reg Loss:       0.001478
    Var Stats:      Min=1.00e-09, Max=4.69e+03, Mean=1.04e+01
    Mean Error L1:  4.3205 m
    Grad Norm:      2233.6616 (Max: 672.2734)
    Dropout:        p1=0.1312, p2=0.1635

--- Validation at iteration 10 ---
  Avg MSE: 42.8159 | Avg ANEES: 43.9609
  >>> New Best Model! (MSE: inf -> 42.8159) <<<
--------------------------------------------------
--- Iter [20/700] (MSE_WARMUP) ---
    Total Loss:     19.9965
    MSE (Metric):   19.9950
    NLL (Metric):   10.1581
    Reg Loss:       0.001475
    Var Stats:      Min=1.00e-09, Max=6.21e+02, Mean=7.92e+00
    Mean Error L1:  2.9326 m
    Grad Norm:      24.0711 (Max: 5.3665)
    Dropout:        p1=0.1303, p2=0.1623

--- Validation at iteration 20 ---
  Avg MSE: 30.5060 | Avg ANEES: 30.8605
  >>> New Best Model! (MSE: 42.8159 -> 30.5060) <<<
--------------------------------------------------
--- Iter [30/700] (MSE_WARMUP) ---
    Total Loss:     16.2093
    MSE (Metric):   16.2078
    NLL (Metric):   12.1276
    Reg Loss:       0.001473
    Var Stats:      Min=1.00e-09, Max=6.49e+03, Mean=9.81e+00
    Mean Error L1:  2.5828 m
    Grad Norm:      12.1287 (Max: 2.2315)
    Dropout:        p1=0.1290, p2=0.1609

--- Validation at iteration 30 ---
  Avg MSE: 33.0821 | Avg ANEES: 30.4980
--------------------------------------------------
--- Iter [40/700] (MSE_WARMUP) ---
    Total Loss:     13.9772
    MSE (Metric):   13.9757
    NLL (Metric):   3.5881
    Reg Loss:       0.001470
    Var Stats:      Min=1.00e-09, Max=4.81e+02, Mean=8.12e+00
    Mean Error L1:  2.4692 m
    Grad Norm:      9.4626 (Max: 1.9800)
    Dropout:        p1=0.1276, p2=0.1593
```

```
--- Validation at iteration 40 ---
  Avg MSE: 19.7804 | Avg ANEES: 23.3173
  >>> New Best Model! (MSE: 30.5060 -> 19.7804) <<<
--------------------------------------------------
--- Iter [50/700] (MSE_WARMUP) ---
    Total Loss:     15.0288
    MSE (Metric):   15.0273
    NLL (Metric):   3.2333
    Reg Loss:       0.001467
    Var Stats:      Min=1.00e-09, Max=1.02e+04, Mean=1.27e+01
    Mean Error L1:  2.4562 m
    Grad Norm:      5.7161 (Max: 1.6315)
    Dropout:        p1=0.1262, p2=0.1577

--- Validation at iteration 50 ---
  Avg MSE: 18.5185 | Avg ANEES: 22.0456
  >>> New Best Model! (MSE: 19.7804 -> 18.5185) <<<
--------------------------------------------------
--- Iter [60/700] (MSE_WARMUP) ---
    Total Loss:     11.6433
    MSE (Metric):   11.6418
    NLL (Metric):   273.2034
    Reg Loss:       0.001465
    Var Stats:      Min=1.00e-09, Max=3.46e+02, Mean=7.91e+00
    Mean Error L1:  2.3083 m
    Grad Norm:      7.1757 (Max: 1.7135)
    Dropout:        p1=0.1248, p2=0.1561

--- Validation at iteration 60 ---
  Avg MSE: 19.1368 | Avg ANEES: 21.9570
--------------------------------------------------
--- Iter [70/700] (MSE_WARMUP) ---
    Total Loss:     12.6634
    MSE (Metric):   12.6620
    NLL (Metric):   4.5053
    Reg Loss:       0.001463
    Var Stats:      Min=1.00e-09, Max=7.79e+03, Mean=1.16e+01
    Mean Error L1:  2.2063 m
    Grad Norm:      6.4857 (Max: 1.7213)
    Dropout:        p1=0.1234, p2=0.1545

--- Validation at iteration 70 ---
  Avg MSE: 18.5297 | Avg ANEES: 20.1597
--------------------------------------------------
--- Iter [80/700] (MSE_WARMUP) ---
    Total Loss:     14.9355
    MSE (Metric):   14.9340
```

```
    NLL (Metric):   6.1585
    Reg Loss:       0.001461
    Var Stats:      Min=1.00e-09, Max=1.37e+04, Mean=1.94e+01
    Mean Error L1:  2.2271 m
    Grad Norm:      6.2420 (Max: 1.0639)
    Dropout:        p1=0.1220, p2=0.1530

--- Validation at iteration 80 ---
  Avg MSE: 14.8728 | Avg ANEES: 22.8797
  >>> New Best Model! (MSE: 18.5185 -> 14.8728) <<<
------------------------------------------------
--- Iter [90/700] (MSE_WARMUP) ---
    Total Loss:     9.8898
    MSE (Metric):   9.8884
    NLL (Metric):   4.5888
    Reg Loss:       0.001459
    Var Stats:      Min=1.00e-09, Max=3.91e+02, Mean=8.49e+00
    Mean Error L1:  2.1557 m
    Grad Norm:      7.6657 (Max: 1.2340)
    Dropout:        p1=0.1207, p2=0.1515

--- Validation at iteration 90 ---
  Avg MSE: 13.5086 | Avg ANEES: 20.9610
  >>> New Best Model! (MSE: 14.8728 -> 13.5086) <<<
------------------------------------------------
--- Iter [100/700] (MSE_WARMUP) ---
    Total Loss:     8.8757
    MSE (Metric):   8.8742
    NLL (Metric):   4.6002
    Reg Loss:       0.001458
    Var Stats:      Min=1.00e-09, Max=3.25e+02, Mean=9.80e+00
    Mean Error L1:  2.0318 m
    Grad Norm:      7.3028 (Max: 1.5017)
    Dropout:        p1=0.1194, p2=0.1500

--- Validation at iteration 100 ---
  Avg MSE: 30.7550 | Avg ANEES: 22.7703
------------------------------------------------
--- Iter [110/700] (MSE_WARMUP) ---
    Total Loss:     9.1972
    MSE (Metric):   9.1958
    NLL (Metric):   3.9669
    Reg Loss:       0.001456
    Var Stats:      Min=1.00e-09, Max=5.48e+02, Mean=8.75e+00
    Mean Error L1:  2.0848 m
    Grad Norm:      5.8003 (Max: 1.0452)
    Dropout:        p1=0.1181, p2=0.1486
```

```
--- Validation at iteration 110 ---
  Avg MSE: 27.0627 | Avg ANEES: 20.6446
------------------------------------------------
--- Iter [120/700] (MSE_WARMUP) ---
    Total Loss:     8.5940
    MSE (Metric):   8.5926
    NLL (Metric):   4.8226
    Reg Loss:       0.001454
    Var Stats:      Min=1.00e-09, Max=3.39e+02, Mean=8.43e+00
    Mean Error L1:  1.9918 m
    Grad Norm:      6.5882 (Max: 1.5069)
    Dropout:        p1=0.1168, p2=0.1473

--- Validation at iteration 120 ---
  Avg MSE: 13.2411 | Avg ANEES: 21.3785
  >>> New Best Model! (MSE: 13.5086 -> 13.2411) <<<
------------------------------------------------
--- Iter [130/700] (MSE_WARMUP) ---
    Total Loss:     8.9927
    MSE (Metric):   8.9912
    NLL (Metric):   5.2115
    Reg Loss:       0.001453
    Var Stats:      Min=1.00e-09, Max=3.23e+02, Mean=8.89e+00
    Mean Error L1:  2.0612 m
    Grad Norm:      6.3762 (Max: 1.1063)
    Dropout:        p1=0.1157, p2=0.1460

--- Validation at iteration 130 ---
  Avg MSE: 23.0068 | Avg ANEES: 22.5467
------------------------------------------------
--- Iter [140/700] (MSE_WARMUP) ---
    Total Loss:     8.0861
    MSE (Metric):   8.0846
    NLL (Metric):   6.1774
    Reg Loss:       0.001451
    Var Stats:      Min=1.00e-09, Max=1.95e+02, Mean=7.88e+00
    Mean Error L1:  1.9483 m
    Grad Norm:      5.9035 (Max: 1.5305)
    Dropout:        p1=0.1147, p2=0.1447

--- Validation at iteration 140 ---
  Avg MSE: 11.8825 | Avg ANEES: 21.6809
  >>> New Best Model! (MSE: 13.2411 -> 11.8825) <<<
------------------------------------------------
--- Iter [150/700] (MSE_WARMUP) ---
    Total Loss:     7.6659
    MSE (Metric):   7.6644
    NLL (Metric):   6.6414
```

```
    Reg Loss:       0.001450
    Var Stats:      Min=1.00e-09, Max=1.94e+02, Mean=7.41e+00
    Mean Error L1:  1.9169 m
    Grad Norm:      6.2821 (Max: 1.3413)
    Dropout:        p1=0.1135, p2=0.1433

--- Validation at iteration 150 ---
  Avg MSE: 10.9476 | Avg ANEES: 29.1083
  >>> New Best Model! (MSE: 11.8825 -> 10.9476) <<<
--------------------------------------------------
--- Iter [160/700] (MSE_WARMUP) ---
    Total Loss:     8.6584
    MSE (Metric):   8.6570
    NLL (Metric):   5.1193
    Reg Loss:       0.001448
    Var Stats:      Min=1.00e-09, Max=2.79e+02, Mean=9.43e+00
    Mean Error L1:  2.0094 m
    Grad Norm:      4.3971 (Max: 0.7270)
    Dropout:        p1=0.1123, p2=0.1420

--- Validation at iteration 160 ---
  Avg MSE: 13.4602 | Avg ANEES: 22.7969
--------------------------------------------------
--- Iter [170/700] (MSE_WARMUP) ---
    Total Loss:     7.9045
    MSE (Metric):   7.9030
    NLL (Metric):   4.4409
    Reg Loss:       0.001446
    Var Stats:      Min=1.00e-09, Max=4.79e+02, Mean=8.17e+00
    Mean Error L1:  1.9233 m
    Grad Norm:      4.5943 (Max: 1.1808)
    Dropout:        p1=0.1111, p2=0.1407

--- Validation at iteration 170 ---
  Avg MSE: 11.7706 | Avg ANEES: 24.3177
--------------------------------------------------
--- Iter [180/700] (MSE_WARMUP) ---
    Total Loss:     6.2697
    MSE (Metric):   6.2682
    NLL (Metric):   6.2738
    Reg Loss:       0.001445
    Var Stats:      Min=1.00e-09, Max=2.31e+02, Mean=8.58e+00
    Mean Error L1:  1.7712 m
    Grad Norm:      3.8417 (Max: 0.8584)
    Dropout:        p1=0.1099, p2=0.1393

--- Validation at iteration 180 ---
  Avg MSE: 12.7715 | Avg ANEES: 24.6249
```

```
------------------------------------------------
--- Iter [190/700] (MSE_WARMUP) ---
    Total Loss:     6.7852
    MSE (Metric):   6.7838
    NLL (Metric):   4.6652
    Reg Loss:       0.001443
    Var Stats:      Min=1.00e-09, Max=2.91e+02, Mean=6.90e+00
    Mean Error L1:  1.8197 m
    Grad Norm:      5.7453 (Max: 1.2576)
    Dropout:        p1=0.1088, p2=0.1379

--- Validation at iteration 190 ---
  Avg MSE: 9.5170 | Avg ANEES: 26.4882
  >>> New Best Model! (MSE: 10.9476 -> 9.5170) <<<
------------------------------------------------
--- Iter [200/700] (MSE_WARMUP) ---
    Total Loss:     6.2786
    MSE (Metric):   6.2771
    NLL (Metric):   31.7730
    Reg Loss:       0.001441
    Var Stats:      Min=1.00e-09, Max=2.35e+02, Mean=6.44e+00
    Mean Error L1:  1.7587 m
    Grad Norm:      4.9644 (Max: 1.0900)
    Dropout:        p1=0.1077, p2=0.1365

--- Validation at iteration 200 ---
  Avg MSE: 15.8843 | Avg ANEES: 25.7946
------------------------------------------------
--- Iter [210/700] (MSE_WARMUP) ---
    Total Loss:     6.7691
    MSE (Metric):   6.7676
    NLL (Metric):   4.9457
    Reg Loss:       0.001440
    Var Stats:      Min=1.00e-09, Max=4.46e+02, Mean=7.25e+00
    Mean Error L1:  1.7971 m
    Grad Norm:      4.9289 (Max: 0.9466)
    Dropout:        p1=0.1067, p2=0.1353

--- Validation at iteration 210 ---
  Avg MSE: 8.9414 | Avg ANEES: 24.3142
  >>> New Best Model! (MSE: 9.5170 -> 8.9414) <<<
------------------------------------------------
--- Iter [220/700] (MSE_WARMUP) ---
    Total Loss:     5.6856
    MSE (Metric):   5.6842
    NLL (Metric):   3.8333
    Reg Loss:       0.001438
    Var Stats:      Min=1.00e-09, Max=1.43e+02, Mean=6.45e+00
```

```
    Mean Error L1:  1.6708 m
    Grad Norm:      4.9921 (Max: 0.9489)
    Dropout:        p1=0.1057, p2=0.1341

--- Validation at iteration 220 ---
  Avg MSE: 10.9647 | Avg ANEES: 28.2317
------------------------------------------------
--- Iter [230/700] (MSE_WARMUP) ---
    Total Loss:     8.2815
    MSE (Metric):   8.2801
    NLL (Metric):   5.4228
    Reg Loss:       0.001437
    Var Stats:      Min=1.00e-09, Max=2.22e+04, Mean=1.39e+01
    Mean Error L1:  1.8168 m
    Grad Norm:      6.1963 (Max: 1.1404)
    Dropout:        p1=0.1047, p2=0.1331

--- Validation at iteration 230 ---
  Avg MSE: 10.2490 | Avg ANEES: 27.7075
------------------------------------------------
--- Iter [240/700] (MSE_WARMUP) ---
    Total Loss:     5.7630
    MSE (Metric):   5.7616
    NLL (Metric):   4.4746
    Reg Loss:       0.001435
    Var Stats:      Min=1.00e-09, Max=3.48e+02, Mean=7.16e+00
    Mean Error L1:  1.6988 m
    Grad Norm:      4.5968 (Max: 0.7769)
    Dropout:        p1=0.1037, p2=0.1320

--- Validation at iteration 240 ---
  Avg MSE: 19.0549 | Avg ANEES: 27.3028
------------------------------------------------
--- Iter [250/700] (MSE_WARMUP) ---
    Total Loss:     6.8438
    MSE (Metric):   6.8424
    NLL (Metric):   5.1685
    Reg Loss:       0.001434
    Var Stats:      Min=1.00e-09, Max=5.39e+03, Mean=9.65e+00
    Mean Error L1:  1.7610 m
    Grad Norm:      4.0428 (Max: 0.8128)
    Dropout:        p1=0.1027, p2=0.1309

--- Validation at iteration 250 ---
  Avg MSE: 20.9177 | Avg ANEES: 29.3029
------------------------------------------------
--- Iter [260/700] (MSE_WARMUP) ---
    Total Loss:     8.5141
```

```
    MSE (Metric):   8.5126
    NLL (Metric):   7.7994
    Reg Loss:       0.001432
    Var Stats:      Min=1.00e-09, Max=1.16e+04, Mean=1.16e+01
    Mean Error L1:  1.8508 m
    Grad Norm:      7.0438 (Max: 1.6685)
    Dropout:        p1=0.1018, p2=0.1298

--- Validation at iteration 260 ---
  Avg MSE: 9.5659 | Avg ANEES: 29.5599
------------------------------------------------
--- Iter [270/700] (MSE_WARMUP) ---
    Total Loss:     6.2952
    MSE (Metric):   6.2937
    NLL (Metric):   4.4113
    Reg Loss:       0.001431
    Var Stats:      Min=1.00e-09, Max=1.93e+02, Mean=6.98e+00
    Mean Error L1:  1.7445 m
    Grad Norm:      4.8638 (Max: 0.9585)
    Dropout:        p1=0.1009, p2=0.1288

--- Validation at iteration 270 ---
  Avg MSE: 9.5223 | Avg ANEES: 31.4456
------------------------------------------------
--- Iter [280/700] (MSE_WARMUP) ---
    Total Loss:     5.8197
    MSE (Metric):   5.8183
    NLL (Metric):   4.1716
    Reg Loss:       0.001430
    Var Stats:      Min=1.00e-09, Max=1.74e+02, Mean=6.09e+00
    Mean Error L1:  1.6888 m
    Grad Norm:      4.7582 (Max: 0.9073)
    Dropout:        p1=0.1000, p2=0.1278

--- Validation at iteration 280 ---
  Avg MSE: 15.2661 | Avg ANEES: 29.7500
------------------------------------------------
--- Iter [290/700] (MSE_WARMUP) ---
    Total Loss:     6.7941
    MSE (Metric):   6.7927
    NLL (Metric):   16.8719
    Reg Loss:       0.001429
    Var Stats:      Min=1.00e-09, Max=6.39e+03, Mean=8.05e+00
    Mean Error L1:  1.7296 m
    Grad Norm:      5.7458 (Max: 1.1834)
    Dropout:        p1=0.0991, p2=0.1269

--- Validation at iteration 290 ---
```

```
  Avg MSE: 14.4187 | Avg ANEES: 29.5534
--------------------------------------------------
--- Iter [300/700] (MSE_WARMUP) ---
    Total Loss:     6.2295
    MSE (Metric):   6.2281
    NLL (Metric):   7.8409
    Reg Loss:       0.001427
    Var Stats:      Min=1.00e-09, Max=1.50e+04, Mean=1.09e+01
    Mean Error L1:  1.6728 m
    Grad Norm:      4.9290 (Max: 0.8736)
    Dropout:        p1=0.0982, p2=0.1259

--- Validation at iteration 300 ---
  Avg MSE: 9.2977 | Avg ANEES: 29.2747
--------------------------------------------------
--- Iter [310/700] (MSE_WARMUP) ---
    Total Loss:     5.6033
    MSE (Metric):   5.6019
    NLL (Metric):   3.9336
    Reg Loss:       0.001427
    Var Stats:      Min=1.00e-09, Max=4.32e+02, Mean=7.22e+00
    Mean Error L1:  1.6572 m
    Grad Norm:      4.3799 (Max: 0.9289)
    Dropout:        p1=0.0973, p2=0.1248

--- Validation at iteration 310 ---
  Avg MSE: 12.7030 | Avg ANEES: 29.9649
--------------------------------------------------
--- Iter [320/700] (MSE_WARMUP) ---
    Total Loss:     5.9119
    MSE (Metric):   5.9105
    NLL (Metric):   9.7297
    Reg Loss:       0.001425
    Var Stats:      Min=1.00e-09, Max=2.34e+02, Mean=6.00e+00
    Mean Error L1:  1.6881 m
    Grad Norm:      3.6907 (Max: 0.6873)
    Dropout:        p1=0.0964, p2=0.1237

--- Validation at iteration 320 ---
  Avg MSE: 15.5041 | Avg ANEES: 33.2373
--------------------------------------------------
--- Iter [330/700] (MSE_WARMUP) ---
    Total Loss:     6.9053
    MSE (Metric):   6.9039
    NLL (Metric):   6.3643
    Reg Loss:       0.001424
    Var Stats:      Min=1.00e-09, Max=1.54e+04, Mean=1.18e+01
    Mean Error L1:  1.7434 m
```

```
    Grad Norm:      4.7489 (Max: 0.9612)
    Dropout:        p1=0.0956, p2=0.1227

--- Validation at iteration 330 ---
  Avg MSE: 19.2436 | Avg ANEES: 40.4991
--------------------------------------------------
--- Iter [340/700] (MSE_WARMUP) ---
    Total Loss:     5.4092
    MSE (Metric):   5.4078
    NLL (Metric):   4.3420
    Reg Loss:       0.001423
    Var Stats:      Min=1.00e-09, Max=2.63e+02, Mean=5.79e+00
    Mean Error L1:  1.6368 m
    Grad Norm:      4.9337 (Max: 1.1576)
    Dropout:        p1=0.0949, p2=0.1219

--- Validation at iteration 340 ---
  Avg MSE: 8.7021 | Avg ANEES: 29.4995
  >>> New Best Model! (MSE: 8.9414 -> 8.7021) <<<
--------------------------------------------------
--- Iter [350/700] (MSE_WARMUP) ---
    Total Loss:     5.8871
    MSE (Metric):   5.8857
    NLL (Metric):   9.4473
    Reg Loss:       0.001422
    Var Stats:      Min=1.00e-09, Max=1.05e+04, Mean=9.32e+00
    Mean Error L1:  1.6389 m
    Grad Norm:      4.1397 (Max: 0.6933)
    Dropout:        p1=0.0941, p2=0.1210

--- Validation at iteration 350 ---
  Avg MSE: 11.8819 | Avg ANEES: 43.5908
--------------------------------------------------
--- Iter [360/700] (MSE_WARMUP) ---
    Total Loss:     5.5999
    MSE (Metric):   5.5985
    NLL (Metric):   4.0418
    Reg Loss:       0.001421
    Var Stats:      Min=1.00e-09, Max=5.79e+02, Mean=6.08e+00
    Mean Error L1:  1.6373 m
    Grad Norm:      4.2143 (Max: 0.9222)
    Dropout:        p1=0.0934, p2=0.1201

--- Validation at iteration 360 ---
  Avg MSE: 9.2830 | Avg ANEES: 36.2105
--------------------------------------------------
--- Iter [370/700] (MSE_WARMUP) ---
    Total Loss:     5.3373
```

```
    MSE (Metric):   5.3359
    NLL (Metric):   6.6990
    Reg Loss:       0.001420
    Var Stats:      Min=1.00e-09, Max=1.87e+02, Mean=4.95e+00
    Mean Error L1:  1.6135 m
    Grad Norm:      5.2531 (Max: 1.0553)
    Dropout:        p1=0.0926, p2=0.1192

--- Validation at iteration 370 ---
  Avg MSE: 11.7380 | Avg ANEES: 32.8740
------------------------------------------------
--- Iter [380/700] (MSE_WARMUP) ---
    Total Loss:     5.5847
    MSE (Metric):   5.5832
    NLL (Metric):   29.5860
    Reg Loss:       0.001419
    Var Stats:      Min=1.00e-09, Max=8.46e+03, Mean=8.61e+00
    Mean Error L1:  1.6053 m
    Grad Norm:      4.1807 (Max: 0.7038)
    Dropout:        p1=0.0918, p2=0.1184

--- Validation at iteration 380 ---
  Avg MSE: 33.4916 | Avg ANEES: 37.4117
------------------------------------------------
--- Iter [390/700] (MSE_WARMUP) ---
    Total Loss:     5.1826
    MSE (Metric):   5.1812
    NLL (Metric):   15.1778
    Reg Loss:       0.001418
    Var Stats:      Min=1.00e-09, Max=1.62e+02, Mean=5.25e+00
    Mean Error L1:  1.5739 m
    Grad Norm:      5.6224 (Max: 1.2067)
    Dropout:        p1=0.0911, p2=0.1176

--- Validation at iteration 390 ---
  Avg MSE: 17.2358 | Avg ANEES: 41.7686
------------------------------------------------
--- Iter [400/700] (MSE_WARMUP) ---
    Total Loss:     4.9641
    MSE (Metric):   4.9627
    NLL (Metric):   6.8829
    Reg Loss:       0.001417
    Var Stats:      Min=1.00e-09, Max=1.95e+02, Mean=5.03e+00
    Mean Error L1:  1.5552 m
    Grad Norm:      7.1495 (Max: 1.2653)
    Dropout:        p1=0.0904, p2=0.1168

--- Validation at iteration 400 ---
```

```
  Avg MSE: 9.4774 | Avg ANEES: 37.6265
--------------------------------------------------
--- Iter [410/700] (MSE_WARMUP) ---
    Total Loss:     5.4026
    MSE (Metric):   5.4012
    NLL (Metric):   19.4496
    Reg Loss:       0.001416
    Var Stats:      Min=1.00e-09, Max=2.66e+02, Mean=5.82e+00
    Mean Error L1:  1.6191 m
    Grad Norm:      5.2411 (Max: 1.0552)
    Dropout:        p1=0.0896, p2=0.1159

--- Validation at iteration 410 ---
  Avg MSE: 8.2496 | Avg ANEES: 39.5536
  >>> New Best Model! (MSE: 8.7021 -> 8.2496) <<<
--------------------------------------------------
--- Iter [420/700] (MSE_WARMUP) ---
    Total Loss:     4.7861
    MSE (Metric):   4.7847
    NLL (Metric):   4.8879
    Reg Loss:       0.001415
    Var Stats:      Min=1.00e-09, Max=1.58e+02, Mean=4.94e+00
    Mean Error L1:  1.5503 m
    Grad Norm:      4.7557 (Max: 1.0263)
    Dropout:        p1=0.0888, p2=0.1151

--- Validation at iteration 420 ---
  Avg MSE: 17.2357 | Avg ANEES: 35.8967
--------------------------------------------------
--- Iter [430/700] (MSE_WARMUP) ---
    Total Loss:     5.1566
    MSE (Metric):   5.1552
    NLL (Metric):   6.8168
    Reg Loss:       0.001414
    Var Stats:      Min=1.00e-09, Max=4.24e+02, Mean=5.18e+00
    Mean Error L1:  1.5824 m
    Grad Norm:      4.7550 (Max: 0.8021)
    Dropout:        p1=0.0880, p2=0.1142

--- Validation at iteration 430 ---
  Avg MSE: 7.9362 | Avg ANEES: 36.3819
  >>> New Best Model! (MSE: 8.2496 -> 7.9362) <<<
--------------------------------------------------
--- Iter [440/700] (MSE_WARMUP) ---
    Total Loss:     5.0658
    MSE (Metric):   5.0644
    NLL (Metric):   9.4614
    Reg Loss:       0.001413
```

```
    Var Stats:      Min=1.00e-09, Max=1.55e+02, Mean=5.12e+00
    Mean Error L1:  1.5781 m
    Grad Norm:      3.9781 (Max: 0.7198)
    Dropout:        p1=0.0872, p2=0.1134

--- Validation at iteration 440 ---
  Avg MSE: 13.7414 | Avg ANEES: 39.9915
--------------------------------------------------
--- Iter [450/700] (MSE_WARMUP) ---
    Total Loss:     4.7392
    MSE (Metric):   4.7378
    NLL (Metric):   8.6966
    Reg Loss:       0.001412
    Var Stats:      Min=1.00e-09, Max=2.86e+02, Mean=5.13e+00
    Mean Error L1:  1.5474 m
    Grad Norm:      4.6074 (Max: 0.8033)
    Dropout:        p1=0.0864, p2=0.1126

--- Validation at iteration 450 ---
  Avg MSE: 8.0302 | Avg ANEES: 37.5913
--------------------------------------------------
--- Iter [460/700] (MSE_WARMUP) ---
    Total Loss:     4.6434
    MSE (Metric):   4.6420
    NLL (Metric):   5.0925
    Reg Loss:       0.001412
    Var Stats:      Min=1.00e-09, Max=1.10e+02, Mean=4.35e+00
    Mean Error L1:  1.5172 m
    Grad Norm:      4.7879 (Max: 1.1951)
    Dropout:        p1=0.0856, p2=0.1117

--- Validation at iteration 460 ---
  Avg MSE: 7.8307 | Avg ANEES: 38.1718
  >>> New Best Model! (MSE: 7.9362 -> 7.8307) <<<
--------------------------------------------------
--- Iter [470/700] (MSE_WARMUP) ---
    Total Loss:     5.1458
    MSE (Metric):   5.1444
    NLL (Metric):   8.0104
    Reg Loss:       0.001411
    Var Stats:      Min=1.00e-09, Max=3.38e+02, Mean=5.23e+00
    Mean Error L1:  1.5887 m
    Grad Norm:      4.0265 (Max: 0.6684)
    Dropout:        p1=0.0847, p2=0.1108

--- Validation at iteration 470 ---
  Avg MSE: 9.5172 | Avg ANEES: 35.3228
--------------------------------------------------
```

```
--- Iter [480/700] (MSE_WARMUP) ---
    Total Loss:     5.2445
    MSE (Metric):   5.2431
    NLL (Metric):   13.5552
    Reg Loss:       0.001409
    Var Stats:      Min=1.00e-09, Max=3.34e+02, Mean=5.31e+00
    Mean Error L1:  1.6094 m
    Grad Norm:      3.4647 (Max: 0.5513)
    Dropout:        p1=0.0839, p2=0.1099

--- Validation at iteration 480 ---
  Avg MSE: 11.0551 | Avg ANEES: 39.8788
--------------------------------------------------
--- Iter [490/700] (MSE_WARMUP) ---
    Total Loss:     5.0534
    MSE (Metric):   5.0520
    NLL (Metric):   7.3110
    Reg Loss:       0.001409
    Var Stats:      Min=1.00e-09, Max=2.24e+02, Mean=5.56e+00
    Mean Error L1:  1.5718 m
    Grad Norm:      4.5773 (Max: 1.2388)
    Dropout:        p1=0.0831, p2=0.1091

--- Validation at iteration 490 ---
  Avg MSE: 6.5921 | Avg ANEES: 40.0340
  >>> New Best Model! (MSE: 7.8307 -> 6.5921) <<<
--------------------------------------------------
--- Iter [500/700] (MSE_WARMUP) ---
    Total Loss:     5.2814
    MSE (Metric):   5.2800
    NLL (Metric):   9.6847
    Reg Loss:       0.001408
    Var Stats:      Min=1.00e-09, Max=1.68e+02, Mean=5.10e+00
    Mean Error L1:  1.6051 m
    Grad Norm:      3.6021 (Max: 0.7939)
    Dropout:        p1=0.0823, p2=0.1083

--- Validation at iteration 500 ---
  Avg MSE: 11.9920 | Avg ANEES: 43.1154
--------------------------------------------------
--- Iter [510/700] (MSE_WARMUP) ---
    Total Loss:     4.7736
    MSE (Metric):   4.7722
    NLL (Metric):   6.2381
    Reg Loss:       0.001407
    Var Stats:      Min=1.00e-09, Max=2.55e+02, Mean=4.73e+00
    Mean Error L1:  1.5240 m
    Grad Norm:      3.1058 (Max: 0.4902)
```

```
    Dropout:        p1=0.0816, p2=0.1074

--- Validation at iteration 510 ---
  Avg MSE: 6.4993 | Avg ANEES: 43.1606
  >>> New Best Model! (MSE: 6.5921 -> 6.4993) <<<
--------------------------------------------------
--- Iter [520/700] (MSE_WARMUP) ---
    Total Loss:     4.5896
    MSE (Metric):   4.5882
    NLL (Metric):   12.0949
    Reg Loss:       0.001406
    Var Stats:      Min=1.00e-09, Max=1.16e+02, Mean=4.64e+00
    Mean Error L1:  1.5135 m
    Grad Norm:      4.7062 (Max: 0.7145)
    Dropout:        p1=0.0810, p2=0.1067

--- Validation at iteration 520 ---
  Avg MSE: 14.7703 | Avg ANEES: 43.8182
--------------------------------------------------
--- Iter [530/700] (MSE_WARMUP) ---
    Total Loss:     4.7716
    MSE (Metric):   4.7702
    NLL (Metric):   5.9336
    Reg Loss:       0.001406
    Var Stats:      Min=1.00e-09, Max=1.41e+02, Mean=4.83e+00
    Mean Error L1:  1.5264 m
    Grad Norm:      3.7644 (Max: 0.5942)
    Dropout:        p1=0.0803, p2=0.1059

--- Validation at iteration 530 ---
  Avg MSE: 12.3606 | Avg ANEES: 3167.6853
--------------------------------------------------
--- Iter [540/700] (MSE_WARMUP) ---
    Total Loss:     4.4820
    MSE (Metric):   4.4806
    NLL (Metric):   66.3846
    Reg Loss:       0.001405
    Var Stats:      Min=1.00e-09, Max=5.68e+02, Mean=5.09e+00
    Mean Error L1:  1.5053 m
    Grad Norm:      4.7973 (Max: 0.8158)
    Dropout:        p1=0.0796, p2=0.1051

--- Validation at iteration 540 ---
  Avg MSE: 12.2498 | Avg ANEES: 43.0401
--------------------------------------------------
--- Iter [550/700] (MSE_WARMUP) ---
    Total Loss:     4.6004
    MSE (Metric):   4.5990
```

```
    NLL (Metric):   9.9877
    Reg Loss:       0.001404
    Var Stats:      Min=1.00e-09, Max=2.45e+02, Mean=4.67e+00
    Mean Error L1:  1.5138 m
    Grad Norm:      5.7008 (Max: 1.0155)
    Dropout:        p1=0.0791, p2=0.1045

--- Validation at iteration 550 ---
  Avg MSE: 15.3360 | Avg ANEES: 36.9722
--------------------------------------------------
--- Iter [560/700] (MSE_WARMUP) ---
    Total Loss:     4.5746
    MSE (Metric):   4.5732
    NLL (Metric):   6.0804
    Reg Loss:       0.001404
    Var Stats:      Min=1.00e-09, Max=1.88e+02, Mean=4.57e+00
    Mean Error L1:  1.5149 m
    Grad Norm:      3.6811 (Max: 0.6456)
    Dropout:        p1=0.0784, p2=0.1038

--- Validation at iteration 560 ---
  Avg MSE: 14.7798 | Avg ANEES: 43.7825
--------------------------------------------------
--- Iter [570/700] (MSE_WARMUP) ---
    Total Loss:     4.4838
    MSE (Metric):   4.4824
    NLL (Metric):   5.5161
    Reg Loss:       0.001403
    Var Stats:      Min=1.00e-09, Max=1.99e+02, Mean=4.44e+00
    Mean Error L1:  1.4957 m
    Grad Norm:      3.0350 (Max: 0.5478)
    Dropout:        p1=0.0778, p2=0.1030

--- Validation at iteration 570 ---
  Avg MSE: 10.5363 | Avg ANEES: 44.9733
--------------------------------------------------
--- Iter [580/700] (MSE_WARMUP) ---
    Total Loss:     4.4858
    MSE (Metric):   4.4844
    NLL (Metric):   5.7859
    Reg Loss:       0.001402
    Var Stats:      Min=1.00e-09, Max=3.18e+02, Mean=4.37e+00
    Mean Error L1:  1.4940 m
    Grad Norm:      3.5723 (Max: 0.7521)
    Dropout:        p1=0.0772, p2=0.1022

--- Validation at iteration 580 ---
  Avg MSE: 7.7525 | Avg ANEES: 40.7628
```

```
------------------------------------------------
--- Iter [590/700] (MSE_WARMUP) ---
    Total Loss:     4.5300
    MSE (Metric):   4.5286
    NLL (Metric):   5.4141
    Reg Loss:       0.001402
    Var Stats:      Min=1.00e-09, Max=2.30e+02, Mean=4.58e+00
    Mean Error L1:  1.4979 m
    Grad Norm:      5.5901 (Max: 0.9086)
    Dropout:        p1=0.0765, p2=0.1015

--- Validation at iteration 590 ---
  Avg MSE: 10.5214 | Avg ANEES: 44.9443
------------------------------------------------
--- Iter [600/700] (MSE_WARMUP) ---
    Total Loss:     4.8365
    MSE (Metric):   4.8351
    NLL (Metric):   5.6403
    Reg Loss:       0.001401
    Var Stats:      Min=1.00e-09, Max=1.65e+02, Mean=4.76e+00
    Mean Error L1:  1.5469 m
    Grad Norm:      4.0806 (Max: 0.8603)
    Dropout:        p1=0.0759, p2=0.1007

--- Validation at iteration 600 ---
  Avg MSE: 7.8157 | Avg ANEES: 41.3247
------------------------------------------------
--- Iter [610/700] (MSE_WARMUP) ---
    Total Loss:     3.8304
    MSE (Metric):   3.8290
    NLL (Metric):   6.3935
    Reg Loss:       0.001401
    Var Stats:      Min=1.00e-09, Max=9.96e+01, Mean=4.07e+00
    Mean Error L1:  1.3911 m
    Grad Norm:      3.3767 (Max: 0.5342)
    Dropout:        p1=0.0752, p2=0.0999

--- Validation at iteration 610 ---
  Avg MSE: 7.9283 | Avg ANEES: 46.3860
------------------------------------------------
--- Iter [620/700] (MSE_WARMUP) ---
    Total Loss:     4.4894
    MSE (Metric):   4.4880
    NLL (Metric):   13.5858
    Reg Loss:       0.001400
    Var Stats:      Min=1.00e-09, Max=1.55e+02, Mean=4.60e+00
    Mean Error L1:  1.4792 m
    Grad Norm:      3.5531 (Max: 0.6067)
```

```
    Dropout:        p1=0.0745, p2=0.0992

--- Validation at iteration 620 ---
  Avg MSE: 10.2172 | Avg ANEES: 44.5305
------------------------------------------------
--- Iter [630/700] (MSE_WARMUP) ---
    Total Loss:     4.3693
    MSE (Metric):   4.3679
    NLL (Metric):   7.4989
    Reg Loss:       0.001399
    Var Stats:      Min=1.00e-09, Max=2.41e+02, Mean=4.38e+00
    Mean Error L1:  1.4899 m
    Grad Norm:      4.3604 (Max: 0.8594)
    Dropout:        p1=0.0739, p2=0.0984

--- Validation at iteration 630 ---
  Avg MSE: 8.2900 | Avg ANEES: 45.5335
------------------------------------------------
--- Iter [640/700] (MSE_WARMUP) ---
    Total Loss:     4.4013
    MSE (Metric):   4.3999
    NLL (Metric):   5.5205
    Reg Loss:       0.001399
    Var Stats:      Min=1.00e-09, Max=1.69e+02, Mean=4.08e+00
    Mean Error L1:  1.4811 m
    Grad Norm:      4.0819 (Max: 0.7793)
    Dropout:        p1=0.0733, p2=0.0978

--- Validation at iteration 640 ---
  Avg MSE: 6.6076 | Avg ANEES: 47.2398
------------------------------------------------
--- Iter [650/700] (MSE_WARMUP) ---
    Total Loss:     4.3702
    MSE (Metric):   4.3688
    NLL (Metric):   7.0640
    Reg Loss:       0.001399
    Var Stats:      Min=1.00e-09, Max=1.69e+02, Mean=4.46e+00
    Mean Error L1:  1.4840 m
    Grad Norm:      3.6247 (Max: 0.7215)
    Dropout:        p1=0.0727, p2=0.0971

--- Validation at iteration 650 ---
  Avg MSE: 6.6039 | Avg ANEES: 54.3028
------------------------------------------------
--- Iter [660/700] (MSE_WARMUP) ---
    Total Loss:     4.6623
    MSE (Metric):   4.6609
    NLL (Metric):   7.4809
```

```
    Reg Loss:       0.001398
    Var Stats:      Min=1.00e-09, Max=3.47e+02, Mean=4.79e+00
    Mean Error L1:  1.5038 m
    Grad Norm:      3.5412 (Max: 0.7158)
    Dropout:        p1=0.0721, p2=0.0964

--- Validation at iteration 660 ---
  Avg MSE: 7.7411 | Avg ANEES: 45.2266
------------------------------------------------
--- Iter [670/700] (MSE_WARMUP) ---
    Total Loss:     4.0035
    MSE (Metric):   4.0021
    NLL (Metric):   39.3704
    Reg Loss:       0.001398
    Var Stats:      Min=1.00e-09, Max=1.54e+02, Mean=4.23e+00
    Mean Error L1:  1.4146 m
    Grad Norm:      5.1353 (Max: 1.0192)
    Dropout:        p1=0.0715, p2=0.0958

--- Validation at iteration 670 ---
  Avg MSE: 7.0500 | Avg ANEES: 43.0074
------------------------------------------------
--- Iter [680/700] (MSE_WARMUP) ---
    Total Loss:     4.6315
    MSE (Metric):   4.6301
    NLL (Metric):   13.7018
    Reg Loss:       0.001397
    Var Stats:      Min=1.00e-09, Max=2.42e+02, Mean=4.70e+00
    Mean Error L1:  1.4990 m
    Grad Norm:      4.2233 (Max: 0.7751)
    Dropout:        p1=0.0710, p2=0.0951

--- Validation at iteration 680 ---
  Avg MSE: 6.5904 | Avg ANEES: 55.7674
------------------------------------------------
--- Iter [690/700] (MSE_WARMUP) ---
    Total Loss:     3.9028
    MSE (Metric):   3.9014
    NLL (Metric):   12.0082
    Reg Loss:       0.001397
    Var Stats:      Min=1.00e-09, Max=1.18e+02, Mean=3.85e+00
    Mean Error L1:  1.3890 m
    Grad Norm:      3.8836 (Max: 0.8457)
    Dropout:        p1=0.0704, p2=0.0945

--- Validation at iteration 690 ---
  Avg MSE: 7.9520 | Avg ANEES: 49.9294
------------------------------------------------
```

```
--- Iter [700/700] (MSE_WARMUP) ---
    Total Loss:     4.2539
    MSE (Metric):   4.2525
    NLL (Metric):   28.6743
    Reg Loss:       0.001396
    Var Stats:      Min=1.00e-09, Max=1.87e+02, Mean=4.39e+00
    Mean Error L1:  1.4449 m
    Grad Norm:      4.0160 (Max: 0.7424)
    Dropout:        p1=0.0699, p2=0.0938

--- Validation at iteration 700 ---
  Avg MSE: 7.9819 | Avg ANEES: 48.6100
------------------------------------------------

Training completed.
Loading best model from iteration 510
  Fáze 1 dokončena. Model uložen do: bkn_curriculum_phase1_len10.pth

============================================================
  START PHASE 2: SeqLen 100 | LR 5e-05 | TBPTT: True
============================================================
  START TBPTT Windowed Training (k=5, w=20)
    Phase 1: MSE Warmup (0 - 1250 updates)
    Phase 2: NLL Optimization (> 1250 updates)
Iter 10 (MSE_WARMUP): Loss 558.6431 | MSE 558.64 | NLL 2.65 | MAE 7.11m
Iter 20 (MSE_WARMUP): Loss 5802.5571 | MSE 5802.56 | NLL 2.69 | MAE 9.24m

--- VALIDATION: MSE 415.77 | ANEES nan ---
  >>> New Best Model! (MSE: 415.77) <<<
----------------------------------------
Iter 30 (MSE_WARMUP): Loss 344.6066 | MSE 344.61 | NLL 2.63 | MAE 6.72m
Iter 40 (MSE_WARMUP): Loss 99.3366 | MSE 99.34 | NLL 2.63 | MAE 6.06m

--- VALIDATION: MSE 471.09 | ANEES nan ---
----------------------------------------
Iter 50 (MSE_WARMUP): Loss 3527.6218 | MSE 3527.62 | NLL 2.71 | MAE 9.44m
Iter 60 (MSE_WARMUP): Loss 276.7864 | MSE 276.78 | NLL 2.67 | MAE 7.07m

--- VALIDATION: MSE 89.35 | ANEES nan ---
  >>> New Best Model! (MSE: 89.35) <<<
----------------------------------------
Iter 70 (MSE_WARMUP): Loss 302.6014 | MSE 302.60 | NLL 2.60 | MAE 6.60m
Iter 80 (MSE_WARMUP): Loss 376.3194 | MSE 376.32 | NLL 2.65 | MAE 6.75m

--- VALIDATION: MSE 135.81 | ANEES nan ---
----------------------------------------
Iter 90 (MSE_WARMUP): Loss 1851.3480 | MSE 1851.35 | NLL 2.67 | MAE 7.91m
Iter 100 (MSE_WARMUP): Loss 2854.8596 | MSE 2854.86 | NLL 2.64 | MAE 9.31m
```

```
--- VALIDATION: MSE 109.41 | ANEES nan ---
----------------------------------------
Iter 110 (MSE_WARMUP): Loss 1289.6458 | MSE 1289.64 | NLL 2.60 | MAE 7.18m
Iter 120 (MSE_WARMUP): Loss 921.4263 | MSE 921.42 | NLL 2.54 | MAE 6.76m

--- VALIDATION: MSE 107.44 | ANEES nan ---
----------------------------------------
Iter 130 (MSE_WARMUP): Loss 331.4363 | MSE 331.43 | NLL 2.75 | MAE 6.57m
Iter 140 (MSE_WARMUP): Loss 2731.9629 | MSE 2731.96 | NLL 2.62 | MAE 8.67m

--- VALIDATION: MSE 88.98 | ANEES nan ---
  >>> New Best Model! (MSE: 88.98) <<<
----------------------------------------
Iter 150 (MSE_WARMUP): Loss 816.1824 | MSE 816.18 | NLL 2.57 | MAE 6.81m
Iter 160 (MSE_WARMUP): Loss 357.4140 | MSE 357.41 | NLL 2.73 | MAE 6.41m

--- VALIDATION: MSE 107.48 | ANEES nan ---
----------------------------------------
Iter 170 (MSE_WARMUP): Loss 749.4650 | MSE 749.46 | NLL 2.50 | MAE 6.95m
Iter 180 (MSE_WARMUP): Loss 523.0220 | MSE 523.02 | NLL 2.70 | MAE 7.23m

--- VALIDATION: MSE 506.18 | ANEES nan ---
----------------------------------------
Iter 190 (MSE_WARMUP): Loss 354.0079 | MSE 354.01 | NLL 2.58 | MAE 6.57m
Iter 200 (MSE_WARMUP): Loss 605.2581 | MSE 605.26 | NLL 2.50 | MAE 7.08m

--- VALIDATION: MSE 50.31 | ANEES nan ---
  >>> New Best Model! (MSE: 50.31) <<<
----------------------------------------
Iter 210 (MSE_WARMUP): Loss 4174.5703 | MSE 4174.57 | NLL 2.59 | MAE 9.08m
Iter 220 (MSE_WARMUP): Loss 1634.7079 | MSE 1634.71 | NLL 2.60 | MAE 7.29m

--- VALIDATION: MSE 127.74 | ANEES nan ---
----------------------------------------
Iter 230 (MSE_WARMUP): Loss 1687.8170 | MSE 1687.82 | NLL 2.71 | MAE 6.96m
Iter 240 (MSE_WARMUP): Loss 180.7703 | MSE 180.77 | NLL 2.64 | MAE 5.98m

--- VALIDATION: MSE 149.14 | ANEES nan ---
----------------------------------------
Iter 250 (MSE_WARMUP): Loss 147.0926 | MSE 147.09 | NLL 2.52 | MAE 6.03m
Iter 260 (MSE_WARMUP): Loss 1671.7738 | MSE 1671.77 | NLL 2.65 | MAE 7.17m

--- VALIDATION: MSE 88.35 | ANEES nan ---
----------------------------------------
Iter 270 (MSE_WARMUP): Loss 167.8230 | MSE 167.82 | NLL 2.69 | MAE 5.48m
Iter 280 (MSE_WARMUP): Loss 2265.8164 | MSE 2265.81 | NLL 2.60 | MAE 8.50m
```

```
--- VALIDATION: MSE 133.81 | ANEES nan ---
----------------------------------------
Iter 290 (MSE_WARMUP): Loss 71.0239 | MSE 71.02 | NLL 2.60 | MAE 5.07m
Iter 300 (MSE_WARMUP): Loss 798.4092 | MSE 798.41 | NLL 2.59 | MAE 7.14m

--- VALIDATION: MSE 55.92 | ANEES nan ---
----------------------------------------
Iter 310 (MSE_WARMUP): Loss 131.1255 | MSE 131.12 | NLL 2.62 | MAE 5.77m
Iter 320 (MSE_WARMUP): Loss 110.9716 | MSE 110.97 | NLL 2.62 | MAE 6.42m

--- VALIDATION: MSE 105.62 | ANEES nan ---
----------------------------------------
Iter 330 (MSE_WARMUP): Loss 169.5529 | MSE 169.55 | NLL 2.70 | MAE 6.08m
Iter 340 (MSE_WARMUP): Loss 1546.2655 | MSE 1546.26 | NLL 2.56 | MAE 7.44m

--- VALIDATION: MSE 110.33 | ANEES nan ---
----------------------------------------
Iter 350 (MSE_WARMUP): Loss 95.2930 | MSE 95.29 | NLL 2.63 | MAE 5.90m
Iter 360 (MSE_WARMUP): Loss 125.1803 | MSE 125.18 | NLL 2.51 | MAE 5.88m

--- VALIDATION: MSE 126.91 | ANEES nan ---
----------------------------------------
Iter 370 (MSE_WARMUP): Loss 80.6536 | MSE 80.65 | NLL 2.66 | MAE 5.39m
Iter 380 (MSE_WARMUP): Loss 1671.3228 | MSE 1671.32 | NLL 2.73 | MAE 7.65m

--- VALIDATION: MSE 109.26 | ANEES nan ---
----------------------------------------
Iter 390 (MSE_WARMUP): Loss 276.7296 | MSE 276.73 | NLL 2.59 | MAE 5.85m
Iter 400 (MSE_WARMUP): Loss 1645.5637 | MSE 1645.56 | NLL 2.60 | MAE 8.12m

--- VALIDATION: MSE 72.00 | ANEES nan ---
----------------------------------------
Iter 410 (MSE_WARMUP): Loss 252.4935 | MSE 252.49 | NLL 2.66 | MAE 5.77m
Iter 420 (MSE_WARMUP): Loss 731.6248 | MSE 731.62 | NLL 2.61 | MAE 6.45m

--- VALIDATION: MSE 82.69 | ANEES nan ---
----------------------------------------
Iter 430 (MSE_WARMUP): Loss 501.7443 | MSE 501.74 | NLL 2.66 | MAE 7.26m
Iter 440 (MSE_WARMUP): Loss 552.4798 | MSE 552.48 | NLL 2.58 | MAE 6.64m

--- VALIDATION: MSE 257.66 | ANEES nan ---
----------------------------------------
Iter 450 (MSE_WARMUP): Loss 459.5245 | MSE 459.52 | NLL 2.64 | MAE 6.10m
Iter 460 (MSE_WARMUP): Loss 73.1921 | MSE 73.19 | NLL 2.79 | MAE 5.17m

--- VALIDATION: MSE 121.01 | ANEES nan ---
----------------------------------------
Iter 470 (MSE_WARMUP): Loss 93.8032 | MSE 93.80 | NLL 2.49 | MAE 5.81m
```

```
Iter 480 (MSE_WARMUP): Loss 194.0559 | MSE 194.05 | NLL 2.54 | MAE 6.06m

--- VALIDATION: MSE 47.40 | ANEES nan ---
  >>> New Best Model! (MSE: 47.40) <<<
----------------------------------------
Iter 490 (MSE_WARMUP): Loss 94.6982 | MSE 94.70 | NLL 2.52 | MAE 5.75m
Iter 500 (MSE_WARMUP): Loss 96.1011 | MSE 96.10 | NLL 2.69 | MAE 5.92m

--- VALIDATION: MSE 63.34 | ANEES nan ---
----------------------------------------
Iter 510 (MSE_WARMUP): Loss 187.1166 | MSE 187.12 | NLL 2.63 | MAE 5.84m
Iter 520 (MSE_WARMUP): Loss 303.4941 | MSE 303.49 | NLL 2.65 | MAE 6.17m

--- VALIDATION: MSE 86.54 | ANEES nan ---
----------------------------------------
Iter 530 (MSE_WARMUP): Loss 449.9598 | MSE 449.96 | NLL 2.58 | MAE 6.26m
Iter 540 (MSE_WARMUP): Loss 246.4913 | MSE 246.49 | NLL 2.65 | MAE 5.97m

--- VALIDATION: MSE 100.50 | ANEES nan ---
----------------------------------------
Iter 550 (MSE_WARMUP): Loss 1402.2594 | MSE 1402.26 | NLL 2.60 | MAE 7.80m
Iter 560 (MSE_WARMUP): Loss 156.4551 | MSE 156.45 | NLL 2.59 | MAE 5.63m

--- VALIDATION: MSE 110.35 | ANEES nan ---
----------------------------------------
Iter 570 (MSE_WARMUP): Loss 110.3724 | MSE 110.37 | NLL 2.54 | MAE 5.54m
Iter 580 (MSE_WARMUP): Loss 88.1565 | MSE 88.15 | NLL 2.69 | MAE 5.61m

--- VALIDATION: MSE 63.35 | ANEES nan ---
----------------------------------------
Iter 590 (MSE_WARMUP): Loss 88.0834 | MSE 88.08 | NLL 2.55 | MAE 5.70m
Iter 600 (MSE_WARMUP): Loss 949.7467 | MSE 949.75 | NLL 2.57 | MAE 7.50m

--- VALIDATION: MSE 96.44 | ANEES nan ---
----------------------------------------
Iter 610 (MSE_WARMUP): Loss 1082.1624 | MSE 1082.16 | NLL 2.52 | MAE 7.64m
Iter 620 (MSE_WARMUP): Loss 102.5235 | MSE 102.52 | NLL 2.60 | MAE 5.29m

--- VALIDATION: MSE 73.63 | ANEES nan ---
----------------------------------------
Iter 630 (MSE_WARMUP): Loss 90.9021 | MSE 90.90 | NLL 2.74 | MAE 5.76m
Iter 640 (MSE_WARMUP): Loss 2184.4431 | MSE 2184.44 | NLL 2.54 | MAE 8.37m

--- VALIDATION: MSE 124.30 | ANEES nan ---
----------------------------------------
Iter 650 (MSE_WARMUP): Loss 334.8435 | MSE 334.84 | NLL 2.69 | MAE 5.86m
Iter 660 (MSE_WARMUP): Loss 2566.6279 | MSE 2566.63 | NLL 2.78 | MAE 8.11m
```

```
--- VALIDATION: MSE 46.51 | ANEES nan ---
  >>> New Best Model! (MSE: 46.51) <<<
----------------------------------------
Iter 670 (MSE_WARMUP): Loss 73.1280 | MSE 73.13 | NLL 2.61 | MAE 5.31m
Iter 680 (MSE_WARMUP): Loss 621.0797 | MSE 621.08 | NLL 2.61 | MAE 7.43m

--- VALIDATION: MSE 239.67 | ANEES nan ---
----------------------------------------
Iter 690 (MSE_WARMUP): Loss 111.8716 | MSE 111.87 | NLL 2.56 | MAE 5.93m
Iter 700 (MSE_WARMUP): Loss 712.4238 | MSE 712.42 | NLL 2.74 | MAE 7.71m

--- VALIDATION: MSE 128.10 | ANEES nan ---
----------------------------------------
Iter 710 (MSE_WARMUP): Loss 71.8266 | MSE 71.83 | NLL 2.57 | MAE 5.08m
Iter 720 (MSE_WARMUP): Loss 206.3193 | MSE 206.32 | NLL 2.52 | MAE 6.37m

--- VALIDATION: MSE 232.27 | ANEES nan ---
----------------------------------------
Iter 730 (MSE_WARMUP): Loss 2431.9382 | MSE 2431.94 | NLL 2.54 | MAE 7.51m
Iter 740 (MSE_WARMUP): Loss 1167.7489 | MSE 1167.75 | NLL 2.64 | MAE 7.78m

--- VALIDATION: MSE 70.79 | ANEES nan ---
----------------------------------------
Iter 750 (MSE_WARMUP): Loss 271.8673 | MSE 271.87 | NLL 2.64 | MAE 6.15m
Iter 760 (MSE_WARMUP): Loss 66.9880 | MSE 66.99 | NLL 2.51 | MAE 5.15m

--- VALIDATION: MSE 226.20 | ANEES nan ---
----------------------------------------
Iter 770 (MSE_WARMUP): Loss 1828.5966 | MSE 1828.59 | NLL 2.75 | MAE 9.15m
Iter 780 (MSE_WARMUP): Loss 2590.3379 | MSE 2590.34 | NLL 2.68 | MAE 7.83m

--- VALIDATION: MSE 76.95 | ANEES nan ---
----------------------------------------
Iter 790 (MSE_WARMUP): Loss 255.0849 | MSE 255.08 | NLL 2.62 | MAE 5.98m
Iter 800 (MSE_WARMUP): Loss 83.3704 | MSE 83.37 | NLL 2.52 | MAE 5.46m

--- VALIDATION: MSE 48.13 | ANEES nan ---
----------------------------------------
Iter 810 (MSE_WARMUP): Loss 198.2743 | MSE 198.27 | NLL 2.72 | MAE 6.00m
Iter 820 (MSE_WARMUP): Loss 1155.7295 | MSE 1155.73 | NLL 2.76 | MAE 7.12m

--- VALIDATION: MSE 53.21 | ANEES nan ---
----------------------------------------
Iter 830 (MSE_WARMUP): Loss 2558.7971 | MSE 2558.80 | NLL 2.65 | MAE 7.96m
Iter 840 (MSE_WARMUP): Loss 84.6636 | MSE 84.66 | NLL 2.59 | MAE 5.41m

--- VALIDATION: MSE 278.46 | ANEES nan ---
----------------------------------------
```

```
Iter 850 (MSE_WARMUP): Loss 313.6930 | MSE 313.69 | NLL 2.57 | MAE 6.75m
Iter 860 (MSE_WARMUP): Loss 2877.1804 | MSE 2877.18 | NLL 2.72 | MAE 8.34m

--- VALIDATION: MSE 225.49 | ANEES nan ---
----------------------------------------
Iter 870 (MSE_WARMUP): Loss 191.2228 | MSE 191.22 | NLL 2.59 | MAE 6.05m
Iter 880 (MSE_WARMUP): Loss 3152.2612 | MSE 3152.26 | NLL 2.65 | MAE 8.22m

--- VALIDATION: MSE 120.70 | ANEES nan ---
----------------------------------------
Iter 890 (MSE_WARMUP): Loss 70.8509 | MSE 70.85 | NLL 2.58 | MAE 5.15m
Iter 900 (MSE_WARMUP): Loss 245.6212 | MSE 245.62 | NLL 2.64 | MAE 5.94m

--- VALIDATION: MSE 58.35 | ANEES nan ---
----------------------------------------
Iter 910 (MSE_WARMUP): Loss 988.8886 | MSE 988.89 | NLL 2.64 | MAE 6.90m
Iter 920 (MSE_WARMUP): Loss 283.6365 | MSE 283.63 | NLL 2.68 | MAE 6.56m

--- VALIDATION: MSE 59.18 | ANEES nan ---
----------------------------------------
Iter 930 (MSE_WARMUP): Loss 1285.3020 | MSE 1285.30 | NLL 2.70 | MAE 8.79m
Iter 940 (MSE_WARMUP): Loss 253.9595 | MSE 253.96 | NLL 2.66 | MAE 7.28m

--- VALIDATION: MSE 98.00 | ANEES nan ---
----------------------------------------
Iter 950 (MSE_WARMUP): Loss 76.6668 | MSE 76.67 | NLL 2.72 | MAE 5.36m
Iter 960 (MSE_WARMUP): Loss 79.5382 | MSE 79.54 | NLL 2.70 | MAE 5.40m

--- VALIDATION: MSE 46.78 | ANEES nan ---
----------------------------------------
Iter 970 (MSE_WARMUP): Loss 3980.2886 | MSE 3980.29 | NLL 2.64 | MAE 8.71m
Iter 980 (MSE_WARMUP): Loss 1431.1041 | MSE 1431.10 | NLL 2.65 | MAE 6.97m

--- VALIDATION: MSE 73.92 | ANEES nan ---
----------------------------------------
Iter 990 (MSE_WARMUP): Loss 88.9687 | MSE 88.97 | NLL 2.81 | MAE 5.43m
Iter 1000 (MSE_WARMUP): Loss 81.4956 | MSE 81.49 | NLL 2.58 | MAE 5.52m

--- VALIDATION: MSE 71.83 | ANEES nan ---
----------------------------------------
Iter 1010 (MSE_WARMUP): Loss 515.9228 | MSE 515.92 | NLL 2.62 | MAE 6.07m
Iter 1020 (MSE_WARMUP): Loss 344.2401 | MSE 344.24 | NLL 2.59 | MAE 6.35m

--- VALIDATION: MSE 125.30 | ANEES nan ---
----------------------------------------
Iter 1030 (MSE_WARMUP): Loss 70.3409 | MSE 70.34 | NLL 2.63 | MAE 5.07m
Iter 1040 (MSE_WARMUP): Loss 116.1745 | MSE 116.17 | NLL 2.62 | MAE 5.69m
```

```
--- VALIDATION: MSE 132.52 | ANEES nan ---
----------------------------------------
Iter 1050 (MSE_WARMUP): Loss 656.0766 | MSE 656.08 | NLL 2.59 | MAE 6.36m
Iter 1060 (MSE_WARMUP): Loss 86.1989 | MSE 86.20 | NLL 2.61 | MAE 5.27m

--- VALIDATION: MSE 47.55 | ANEES nan ---
----------------------------------------
Iter 1070 (MSE_WARMUP): Loss 154.4938 | MSE 154.49 | NLL 2.72 | MAE 5.81m
Iter 1080 (MSE_WARMUP): Loss 81.8128 | MSE 81.81 | NLL 2.65 | MAE 5.52m

--- VALIDATION: MSE 79.58 | ANEES nan ---
----------------------------------------
Iter 1090 (MSE_WARMUP): Loss 682.8953 | MSE 682.89 | NLL 2.63 | MAE 7.41m
Iter 1100 (MSE_WARMUP): Loss 93.0371 | MSE 93.04 | NLL 2.52 | MAE 5.95m

--- VALIDATION: MSE 105.09 | ANEES nan ---
----------------------------------------
Iter 1110 (MSE_WARMUP): Loss 437.0875 | MSE 437.09 | NLL 2.76 | MAE 7.15m
Iter 1120 (MSE_WARMUP): Loss 76.8480 | MSE 76.85 | NLL 2.79 | MAE 5.28m

--- VALIDATION: MSE 71.17 | ANEES nan ---
----------------------------------------
Iter 1130 (MSE_WARMUP): Loss 110.1040 | MSE 110.10 | NLL 2.68 | MAE 5.29m
Iter 1140 (MSE_WARMUP): Loss 3238.7812 | MSE 3238.78 | NLL 2.78 | MAE 9.03m

--- VALIDATION: MSE 72.19 | ANEES nan ---
----------------------------------------
Iter 1150 (MSE_WARMUP): Loss 80.1048 | MSE 80.10 | NLL 2.71 | MAE 5.42m
Iter 1160 (MSE_WARMUP): Loss 85.3074 | MSE 85.31 | NLL 2.69 | MAE 5.45m

--- VALIDATION: MSE 65.33 | ANEES nan ---
----------------------------------------
Iter 1170 (MSE_WARMUP): Loss 79.6796 | MSE 79.68 | NLL 2.59 | MAE 5.43m
Iter 1180 (MSE_WARMUP): Loss 1536.3307 | MSE 1536.33 | NLL 2.89 | MAE 7.96m

--- VALIDATION: MSE 105.66 | ANEES nan ---
----------------------------------------
Iter 1190 (MSE_WARMUP): Loss 135.6897 | MSE 135.69 | NLL 2.74 | MAE 5.41m
Iter 1200 (MSE_WARMUP): Loss 83.7496 | MSE 83.75 | NLL 2.73 | MAE 5.48m

--- VALIDATION: MSE 95.51 | ANEES nan ---
----------------------------------------
Iter 1210 (MSE_WARMUP): Loss 242.6692 | MSE 242.67 | NLL 2.65 | MAE 6.00m
Iter 1220 (MSE_WARMUP): Loss 705.7354 | MSE 705.73 | NLL 2.78 | MAE 7.45m

--- VALIDATION: MSE 51.85 | ANEES nan ---
----------------------------------------
Iter 1230 (MSE_WARMUP): Loss 5863.3760 | MSE 5863.37 | NLL 2.68 | MAE 9.87m
```

```
Iter 1240 (MSE_WARMUP): Loss 122.9403 | MSE 122.94 | NLL 2.77 | MAE 5.68m

--- VALIDATION: MSE 101.05 | ANEES nan ---
----------------------------------------
Iter 1250 (MSE_WARMUP): Loss 106.1423 | MSE 106.14 | NLL 2.66 | MAE 6.06m
Iter 1260 (NLL_OPTIM): Loss 101334.6172 | MSE 1013.32 | NLL 2.76 | MAE 7.00m

--- VALIDATION: MSE 98.36 | ANEES nan ---
----------------------------------------
Iter 1270 (NLL_OPTIM): Loss 8322.1865 | MSE 83.19 | NLL 2.72 | MAE 5.51m
Iter 1280 (NLL_OPTIM): Loss 28124.5977 | MSE 281.22 | NLL 2.70 | MAE 6.06m

--- VALIDATION: MSE 199.21 | ANEES nan ---
----------------------------------------
Iter 1290 (NLL_OPTIM): Loss 57169.8281 | MSE 571.67 | NLL 2.59 | MAE 6.64m
Iter 1300 (NLL_OPTIM): Loss 538048.0000 | MSE 5380.45 | NLL 2.83 | MAE 8.90m

--- VALIDATION: MSE 224.73 | ANEES nan ---
----------------------------------------
Iter 1310 (NLL_OPTIM): Loss 62781.3203 | MSE 627.79 | NLL 2.78 | MAE 6.15m
Iter 1320 (NLL_OPTIM): Loss 22601.1836 | MSE 225.99 | NLL 2.56 | MAE 5.98m

--- VALIDATION: MSE 127.96 | ANEES nan ---
----------------------------------------
Iter 1330 (NLL_OPTIM): Loss 229718.0312 | MSE 2297.15 | NLL 2.70 | MAE 8.32m
Iter 1340 (NLL_OPTIM): Loss 8168.4131 | MSE 81.66 | NLL 2.56 | MAE 5.37m

--- VALIDATION: MSE 44.09 | ANEES nan ---
  >>> New Best Model! (MSE: 44.09) <<<
----------------------------------------
Iter 1350 (NLL_OPTIM): Loss 346991.4688 | MSE 3469.89 | NLL 2.67 | MAE 7.99m
Iter 1360 (NLL_OPTIM): Loss 8261.9873 | MSE 82.59 | NLL 2.60 | MAE 5.52m

--- VALIDATION: MSE 118.09 | ANEES nan ---
----------------------------------------
Iter 1370 (NLL_OPTIM): Loss 28187.7266 | MSE 281.85 | NLL 2.67 | MAE 5.89m
Iter 1380 (NLL_OPTIM): Loss 267446.0312 | MSE 2674.43 | NLL 2.72 | MAE 9.43m

--- VALIDATION: MSE 59.66 | ANEES nan ---
----------------------------------------
Iter 1390 (NLL_OPTIM): Loss 26197.3730 | MSE 261.95 | NLL 2.71 | MAE 5.52m
Iter 1400 (NLL_OPTIM): Loss 34565.3242 | MSE 345.63 | NLL 2.61 | MAE 6.66m

--- VALIDATION: MSE 228.39 | ANEES nan ---
----------------------------------------
Iter 1410 (NLL_OPTIM): Loss 7536.0430 | MSE 75.33 | NLL 2.62 | MAE 5.18m
Iter 1420 (NLL_OPTIM): Loss 17487.9590 | MSE 174.85 | NLL 2.63 | MAE 5.85m
```

```
--- VALIDATION: MSE 58.61 | ANEES nan ---
----------------------------------------
Iter 1430 (NLL_OPTIM): Loss 63074.4492 | MSE 630.72 | NLL 2.73 | MAE 6.72m
Iter 1440 (NLL_OPTIM): Loss 8136.7583 | MSE 81.34 | NLL 2.73 | MAE 5.53m

--- VALIDATION: MSE 50.61 | ANEES nan ---
----------------------------------------
Iter 1450 (NLL_OPTIM): Loss 164193.4062 | MSE 1641.91 | NLL 2.63 | MAE 7.13m
Iter 1460 (NLL_OPTIM): Loss 25981.9883 | MSE 259.79 | NLL 2.66 | MAE 5.50m

--- VALIDATION: MSE 81.70 | ANEES nan ---
----------------------------------------
Iter 1470 (NLL_OPTIM): Loss 17654.6582 | MSE 176.52 | NLL 2.58 | MAE 5.84m
Iter 1480 (NLL_OPTIM): Loss 7657.0977 | MSE 76.54 | NLL 2.61 | MAE 5.29m

--- VALIDATION: MSE 45.75 | ANEES nan ---
----------------------------------------
Iter 1490 (NLL_OPTIM): Loss 20464.3828 | MSE 204.62 | NLL 2.56 | MAE 5.34m
Iter 1500 (NLL_OPTIM): Loss 604275.2500 | MSE 6042.73 | NLL 2.69 | MAE 9.85m

--- VALIDATION: MSE 257.54 | ANEES nan ---
----------------------------------------
Iter 1510 (NLL_OPTIM): Loss 9776.6113 | MSE 97.74 | NLL 2.69 | MAE 5.41m
Iter 1520 (NLL_OPTIM): Loss 267647.1875 | MSE 2676.45 | NLL 2.69 | MAE 9.13m

--- VALIDATION: MSE 65.63 | ANEES nan ---
----------------------------------------
Iter 1530 (NLL_OPTIM): Loss 35921.4570 | MSE 359.19 | NLL 2.63 | MAE 5.99m
Iter 1540 (NLL_OPTIM): Loss 14471.3086 | MSE 144.69 | NLL 2.67 | MAE 5.46m

--- VALIDATION: MSE 128.71 | ANEES nan ---
----------------------------------------
Iter 1550 (NLL_OPTIM): Loss 7857.6816 | MSE 78.55 | NLL 2.72 | MAE 5.47m
Iter 1560 (NLL_OPTIM): Loss 32231.6504 | MSE 322.29 | NLL 2.67 | MAE 6.40m

--- VALIDATION: MSE 49.11 | ANEES nan ---
----------------------------------------
Iter 1570 (NLL_OPTIM): Loss 16657.3770 | MSE 166.55 | NLL 2.79 | MAE 5.74m
Iter 1580 (NLL_OPTIM): Loss 21205.5684 | MSE 212.03 | NLL 2.54 | MAE 5.90m

--- VALIDATION: MSE 229.10 | ANEES nan ---
----------------------------------------
Iter 1590 (NLL_OPTIM): Loss 13871.3125 | MSE 138.69 | NLL 2.79 | MAE 5.28m
Iter 1600 (NLL_OPTIM): Loss 7770.6719 | MSE 77.68 | NLL 2.66 | MAE 5.37m

--- VALIDATION: MSE 45.55 | ANEES nan ---
----------------------------------------
Iter 1610 (NLL_OPTIM): Loss 13602.6270 | MSE 136.00 | NLL 2.58 | MAE 5.49m
```

```
Iter 1620 (NLL_OPTIM): Loss 56113.6016 | MSE 561.11 | NLL 2.74 | MAE 6.48m

--- VALIDATION: MSE 49.55 | ANEES nan ---
----------------------------------------
Iter 1630 (NLL_OPTIM): Loss 134072.7656 | MSE 1340.70 | NLL 2.69 | MAE 7.59m
Iter 1640 (NLL_OPTIM): Loss 7406.0161 | MSE 74.03 | NLL 2.69 | MAE 5.27m

--- VALIDATION: MSE 108.04 | ANEES nan ---
----------------------------------------
Iter 1650 (NLL_OPTIM): Loss 10525.9346 | MSE 105.23 | NLL 2.71 | MAE 5.73m
Iter 1660 (NLL_OPTIM): Loss 38376.4688 | MSE 383.74 | NLL 2.67 | MAE 6.59m

--- VALIDATION: MSE 56.15 | ANEES nan ---
----------------------------------------
Iter 1670 (NLL_OPTIM): Loss 74221.3516 | MSE 742.19 | NLL 2.67 | MAE 6.96m
Iter 1680 (NLL_OPTIM): Loss 12762.6152 | MSE 127.60 | NLL 2.70 | MAE 5.28m

--- VALIDATION: MSE 888.10 | ANEES nan ---
----------------------------------------
Iter 1690 (NLL_OPTIM): Loss 17304.0469 | MSE 173.01 | NLL 2.61 | MAE 5.98m
Iter 1700 (NLL_OPTIM): Loss 7564.5376 | MSE 75.62 | NLL 2.72 | MAE 5.30m

--- VALIDATION: MSE 45.87 | ANEES nan ---
----------------------------------------
Iter 1710 (NLL_OPTIM): Loss 130160.1953 | MSE 1301.58 | NLL 2.60 | MAE 7.66m
Iter 1720 (NLL_OPTIM): Loss 7705.1572 | MSE 77.03 | NLL 2.62 | MAE 5.38m

--- VALIDATION: MSE 69.89 | ANEES nan ---
----------------------------------------
Iter 1730 (NLL_OPTIM): Loss 176141.8438 | MSE 1761.39 | NLL 2.75 | MAE 8.23m
Iter 1740 (NLL_OPTIM): Loss 23694.0137 | MSE 236.91 | NLL 2.63 | MAE 7.10m

--- VALIDATION: MSE 108.32 | ANEES nan ---
----------------------------------------
Iter 1750 (NLL_OPTIM): Loss 7781.6641 | MSE 77.79 | NLL 2.58 | MAE 5.37m
Iter 1760 (NLL_OPTIM): Loss 13606.6562 | MSE 136.04 | NLL 2.77 | MAE 5.32m

--- VALIDATION: MSE 174.06 | ANEES nan ---
----------------------------------------
Iter 1770 (NLL_OPTIM): Loss 7165.8726 | MSE 71.63 | NLL 2.65 | MAE 5.18m
Iter 1780 (NLL_OPTIM): Loss 18731.1191 | MSE 187.29 | NLL 2.56 | MAE 5.55m

--- VALIDATION: MSE 63.37 | ANEES nan ---
----------------------------------------
Iter 1790 (NLL_OPTIM): Loss 252742.4688 | MSE 2527.40 | NLL 2.59 | MAE 8.30m
Iter 1800 (NLL_OPTIM): Loss 36370.6328 | MSE 363.68 | NLL 2.69 | MAE 6.27m

--- VALIDATION: MSE 46.22 | ANEES nan ---
```

```
----------------------------------------
Iter 1810 (NLL_OPTIM): Loss 39551.4219 | MSE 395.49 | NLL 2.57 | MAE 6.80m
Iter 1820 (NLL_OPTIM): Loss 21831.6816 | MSE 218.29 | NLL 2.75 | MAE 6.08m

--- VALIDATION: MSE 58.84 | ANEES nan ---
----------------------------------------
Iter 1830 (NLL_OPTIM): Loss 7338.9302 | MSE 73.36 | NLL 2.75 | MAE 5.18m
Iter 1840 (NLL_OPTIM): Loss 41074.6523 | MSE 410.72 | NLL 2.68 | MAE 6.30m

--- VALIDATION: MSE 81.45 | ANEES nan ---
----------------------------------------
Iter 1850 (NLL_OPTIM): Loss 104383.9922 | MSE 1043.81 | NLL 2.66 | MAE 7.05m
Iter 1860 (NLL_OPTIM): Loss 7869.3223 | MSE 78.67 | NLL 2.62 | MAE 5.32m

--- VALIDATION: MSE 63.36 | ANEES nan ---
----------------------------------------
Iter 1870 (NLL_OPTIM): Loss 6826.7661 | MSE 68.24 | NLL 2.87 | MAE 5.03m
Iter 1880 (NLL_OPTIM): Loss 303572.1875 | MSE 3035.69 | NLL 2.79 | MAE 8.64m

--- VALIDATION: MSE 47.49 | ANEES nan ---
----------------------------------------
Iter 1890 (NLL_OPTIM): Loss 9961.9160 | MSE 99.59 | NLL 2.74 | MAE 5.52m
Iter 1900 (NLL_OPTIM): Loss 45301.5859 | MSE 452.99 | NLL 2.79 | MAE 6.47m

--- VALIDATION: MSE 75.65 | ANEES nan ---
----------------------------------------
Iter 1910 (NLL_OPTIM): Loss 481533.2188 | MSE 4815.31 | NLL 2.64 | MAE 10.75m
Iter 1920 (NLL_OPTIM): Loss 8070.0713 | MSE 80.67 | NLL 2.70 | MAE 5.40m

--- VALIDATION: MSE 252.20 | ANEES nan ---
----------------------------------------
Iter 1930 (NLL_OPTIM): Loss 22443.5977 | MSE 224.41 | NLL 2.70 | MAE 5.81m
Iter 1940 (NLL_OPTIM): Loss 7577.5664 | MSE 75.75 | NLL 2.68 | MAE 5.22m

--- VALIDATION: MSE 98.59 | ANEES nan ---
----------------------------------------
Iter 1950 (NLL_OPTIM): Loss 8403.0430 | MSE 84.01 | NLL 2.53 | MAE 5.50m
Iter 1960 (NLL_OPTIM): Loss 27384.6113 | MSE 273.82 | NLL 2.69 | MAE 6.23m

--- VALIDATION: MSE 132.45 | ANEES nan ---
----------------------------------------
Iter 1970 (NLL_OPTIM): Loss 152359.7969 | MSE 1523.57 | NLL 2.67 | MAE 7.40m
Iter 1980 (NLL_OPTIM): Loss 34810.1953 | MSE 348.07 | NLL 2.80 | MAE 6.22m

--- VALIDATION: MSE 119.52 | ANEES nan ---
----------------------------------------
Iter 1990 (NLL_OPTIM): Loss 53912.3984 | MSE 539.10 | NLL 2.69 | MAE 6.88m
Iter 2000 (NLL_OPTIM): Loss 8249.8359 | MSE 82.47 | NLL 2.66 | MAE 5.48m
```

```
--- VALIDATION: MSE 97.28 | ANEES nan ---
----------------------------------------
Iter 2010 (NLL_OPTIM): Loss 37781.1328 | MSE 377.79 | NLL 2.62 | MAE 6.02m
Iter 2020 (NLL_OPTIM): Loss 59484.6719 | MSE 594.82 | NLL 2.56 | MAE 6.62m

--- VALIDATION: MSE 98.90 | ANEES nan ---
----------------------------------------
Iter 2030 (NLL_OPTIM): Loss 379550.3438 | MSE 3795.47 | NLL 2.85 | MAE 9.26m
Iter 2040 (NLL_OPTIM): Loss 8093.0610 | MSE 80.90 | NLL 2.66 | MAE 5.31m

--- VALIDATION: MSE 110.22 | ANEES nan ---
----------------------------------------
Iter 2050 (NLL_OPTIM): Loss 108467.3828 | MSE 1084.65 | NLL 2.63 | MAE 7.04m
Iter 2060 (NLL_OPTIM): Loss 27491.7031 | MSE 274.89 | NLL 2.75 | MAE 5.92m

--- VALIDATION: MSE 46.81 | ANEES nan ---
----------------------------------------
Iter 2070 (NLL_OPTIM): Loss 224623.2656 | MSE 2246.21 | NLL 2.64 | MAE 7.58m
Iter 2080 (NLL_OPTIM): Loss 7302.2212 | MSE 73.00 | NLL 2.58 | MAE 5.14m

--- VALIDATION: MSE 46.42 | ANEES nan ---
----------------------------------------
Iter 2090 (NLL_OPTIM): Loss 9412.9873 | MSE 94.10 | NLL 2.52 | MAE 5.38m
Iter 2100 (NLL_OPTIM): Loss 7571.0884 | MSE 75.68 | NLL 2.65 | MAE 5.28m

--- VALIDATION: MSE 119.39 | ANEES nan ---
----------------------------------------
Iter 2110 (NLL_OPTIM): Loss 233052.2188 | MSE 2330.50 | NLL 2.64 | MAE 7.33m
Iter 2120 (NLL_OPTIM): Loss 7644.6665 | MSE 76.42 | NLL 2.70 | MAE 5.40m

--- VALIDATION: MSE 100.36 | ANEES nan ---
----------------------------------------
Iter 2130 (NLL_OPTIM): Loss 47315.0273 | MSE 473.12 | NLL 2.55 | MAE 6.52m
Iter 2140 (NLL_OPTIM): Loss 508133.0938 | MSE 5081.30 | NLL 2.69 | MAE 8.81m

--- VALIDATION: MSE 97.53 | ANEES nan ---
----------------------------------------
Iter 2150 (NLL_OPTIM): Loss 8319.4756 | MSE 83.17 | NLL 2.66 | MAE 5.47m
Iter 2160 (NLL_OPTIM): Loss 11955.7236 | MSE 119.53 | NLL 2.57 | MAE 5.90m

--- VALIDATION: MSE 100.17 | ANEES nan ---
----------------------------------------
Iter 2170 (NLL_OPTIM): Loss 48143.3906 | MSE 481.41 | NLL 2.71 | MAE 6.44m
Iter 2180 (NLL_OPTIM): Loss 32268.3047 | MSE 322.66 | NLL 2.62 | MAE 5.85m

--- VALIDATION: MSE 167.52 | ANEES nan ---
----------------------------------------
```

```
Iter 2190 (NLL_OPTIM): Loss 139902.9062 | MSE 1399.00 | NLL 2.77 | MAE 6.96m
Iter 2200 (NLL_OPTIM): Loss 27037.3984 | MSE 270.35 | NLL 2.59 | MAE 5.66m

--- VALIDATION: MSE 143.70 | ANEES nan ---
----------------------------------------
Iter 2210 (NLL_OPTIM): Loss 22499.3223 | MSE 224.97 | NLL 2.72 | MAE 6.07m
Iter 2220 (NLL_OPTIM): Loss 370109.5000 | MSE 3701.07 | NLL 2.64 | MAE 8.81m

--- VALIDATION: MSE 129.53 | ANEES nan ---
----------------------------------------
Iter 2230 (NLL_OPTIM): Loss 37102.8125 | MSE 371.00 | NLL 2.67 | MAE 6.18m
Iter 2240 (NLL_OPTIM): Loss 7045.1831 | MSE 70.43 | NLL 2.59 | MAE 5.12m

--- VALIDATION: MSE 733.57 | ANEES nan ---
----------------------------------------
Iter 2250 (NLL_OPTIM): Loss 37733.0195 | MSE 377.30 | NLL 2.69 | MAE 6.58m
Iter 2260 (NLL_OPTIM): Loss 25682.9961 | MSE 256.80 | NLL 2.61 | MAE 6.34m

--- VALIDATION: MSE 126.63 | ANEES nan ---
----------------------------------------
Iter 2270 (NLL_OPTIM): Loss 16914.3848 | MSE 169.12 | NLL 2.62 | MAE 5.61m
Iter 2280 (NLL_OPTIM): Loss 8015.8921 | MSE 80.13 | NLL 2.56 | MAE 5.48m

--- VALIDATION: MSE 163.98 | ANEES nan ---
----------------------------------------
Iter 2290 (NLL_OPTIM): Loss 8205.0791 | MSE 82.02 | NLL 2.65 | MAE 5.57m
Iter 2300 (NLL_OPTIM): Loss 7781.8271 | MSE 77.79 | NLL 2.75 | MAE 5.37m

--- VALIDATION: MSE 45.99 | ANEES nan ---
----------------------------------------
Iter 2310 (NLL_OPTIM): Loss 8982.4199 | MSE 89.80 | NLL 2.61 | MAE 5.66m
Iter 2320 (NLL_OPTIM): Loss 84918.6250 | MSE 849.16 | NLL 2.72 | MAE 7.07m

--- VALIDATION: MSE 224.96 | ANEES nan ---
----------------------------------------
Iter 2330 (NLL_OPTIM): Loss 78882.3750 | MSE 788.80 | NLL 2.83 | MAE 7.87m
Iter 2340 (NLL_OPTIM): Loss 14839.3984 | MSE 148.37 | NLL 2.84 | MAE 5.73m

--- VALIDATION: MSE 96.52 | ANEES nan ---
----------------------------------------
Iter 2350 (NLL_OPTIM): Loss 24167.9766 | MSE 241.65 | NLL 2.61 | MAE 6.20m
Iter 2360 (NLL_OPTIM): Loss 60637.0078 | MSE 606.34 | NLL 2.77 | MAE 6.60m

--- VALIDATION: MSE 452.05 | ANEES nan ---
----------------------------------------
Iter 2370 (NLL_OPTIM): Loss 19877.7656 | MSE 198.75 | NLL 2.56 | MAE 5.81m
Iter 2380 (NLL_OPTIM): Loss 9002.3281 | MSE 90.00 | NLL 2.72 | MAE 5.64m
```

```
--- VALIDATION: MSE 91.26 | ANEES nan ---
----------------------------------------
Iter 2390 (NLL_OPTIM): Loss 227921.2500 | MSE 2279.19 | NLL 2.72 | MAE 7.65m
Iter 2400 (NLL_OPTIM): Loss 89025.9531 | MSE 890.23 | NLL 2.66 | MAE 7.66m

--- VALIDATION: MSE 487.44 | ANEES nan ---
----------------------------------------
Iter 2410 (NLL_OPTIM): Loss 18532.8633 | MSE 185.30 | NLL 2.57 | MAE 5.99m
Iter 2420 (NLL_OPTIM): Loss 157033.5781 | MSE 1570.31 | NLL 2.62 | MAE 7.07m

--- VALIDATION: MSE 192.78 | ANEES nan ---
----------------------------------------
Iter 2430 (NLL_OPTIM): Loss 388196.0625 | MSE 3881.93 | NLL 2.79 | MAE 9.84m
Iter 2440 (NLL_OPTIM): Loss 17054.5254 | MSE 170.52 | NLL 2.68 | MAE 5.65m

--- VALIDATION: MSE 236.55 | ANEES nan ---
----------------------------------------
Iter 2450 (NLL_OPTIM): Loss 13449.8369 | MSE 134.47 | NLL 2.56 | MAE 5.54m
Iter 2460 (NLL_OPTIM): Loss 9893.8584 | MSE 98.91 | NLL 2.82 | MAE 5.48m

--- VALIDATION: MSE 224.50 | ANEES nan ---
----------------------------------------
Iter 2470 (NLL_OPTIM): Loss 25030.1797 | MSE 250.27 | NLL 2.78 | MAE 6.15m
Iter 2480 (NLL_OPTIM): Loss 7499.6841 | MSE 74.97 | NLL 2.61 | MAE 5.35m

--- VALIDATION: MSE 53.58 | ANEES nan ---
----------------------------------------
Iter 2490 (NLL_OPTIM): Loss 58743.0469 | MSE 587.40 | NLL 2.67 | MAE 6.77m
Iter 2500 (NLL_OPTIM): Loss 14150.1416 | MSE 141.47 | NLL 2.67 | MAE 5.68m

--- VALIDATION: MSE 111.68 | ANEES nan ---
----------------------------------------
Training completed.
  Fáze 2 dokončena. Model uložen do: bkn_curriculum_phase2_len100.pth

============================================================
  START PHASE 3: SeqLen 300 | LR 1e-06 | TBPTT: True
============================================================
  START TBPTT Windowed Training (k=5, w=20)
    Phase 1: MSE Warmup (0 - 500 updates)
    Phase 2: NLL Optimization (> 500 updates)
Iter 10 (MSE_WARMUP): Loss 143.7911 | MSE 143.79 | NLL 2.62 | MAE 6.99m
Iter 20 (MSE_WARMUP): Loss 73.2724 | MSE 73.27 | NLL 2.69 | MAE 5.32m

--- VALIDATION: MSE 1252.71 | ANEES nan ---
  >>> New Best Model! (MSE: 1252.71) <<<
----------------------------------------
Iter 30 (MSE_WARMUP): Loss 348.2803 | MSE 348.28 | NLL 2.95 | MAE 9.52m
```

```
Iter 40 (MSE_WARMUP): Loss 244.4146 | MSE 244.41 | NLL 2.77 | MAE 8.38m

--- VALIDATION: MSE 3059.42 | ANEES nan ---
----------------------------------------
Iter 50 (MSE_WARMUP): Loss 17129.7812 | MSE 17129.78 | NLL 2.63 | MAE 12.02m
Iter 60 (MSE_WARMUP): Loss 5316.1963 | MSE 5316.19 | NLL 2.81 | MAE 15.39m

--- VALIDATION: MSE 336.03 | ANEES nan ---
  >>> New Best Model! (MSE: 336.03) <<<
----------------------------------------
Iter 70 (MSE_WARMUP): Loss 186.6680 | MSE 186.67 | NLL 2.93 | MAE 8.27m
Iter 80 (MSE_WARMUP): Loss 331.5533 | MSE 331.55 | NLL 2.75 | MAE 6.20m

--- VALIDATION: MSE 331.44 | ANEES nan ---
  >>> New Best Model! (MSE: 331.44) <<<
----------------------------------------
Iter 90 (MSE_WARMUP): Loss 1609.8483 | MSE 1609.85 | NLL 2.85 | MAE 11.80m
Iter 100 (MSE_WARMUP): Loss 157.3519 | MSE 157.35 | NLL 2.83 | MAE 7.44m

--- VALIDATION: MSE 281.93 | ANEES nan ---
  >>> New Best Model! (MSE: 281.93) <<<
----------------------------------------
Iter 110 (MSE_WARMUP): Loss 75.0027 | MSE 75.00 | NLL 2.55 | MAE 5.38m
Iter 120 (MSE_WARMUP): Loss 1092.2246 | MSE 1092.22 | NLL 2.81 | MAE 11.02m

--- VALIDATION: MSE 523.85 | ANEES nan ---
----------------------------------------
Iter 130 (MSE_WARMUP): Loss 342.8843 | MSE 342.88 | NLL 2.81 | MAE 8.28m
Iter 140 (MSE_WARMUP): Loss 74.1962 | MSE 74.19 | NLL 2.79 | MAE 5.16m

--- VALIDATION: MSE 346.75 | ANEES nan ---
----------------------------------------
Iter 150 (MSE_WARMUP): Loss 1386.8528 | MSE 1386.85 | NLL 2.75 | MAE 11.20m
Iter 160 (MSE_WARMUP): Loss 533.2346 | MSE 533.23 | NLL 2.73 | MAE 8.73m

--- VALIDATION: MSE 1567.75 | ANEES nan ---
----------------------------------------
Iter 170 (MSE_WARMUP): Loss 3266.5432 | MSE 3266.54 | NLL 2.66 | MAE 9.17m
Iter 180 (MSE_WARMUP): Loss 20995.1367 | MSE 20995.13 | NLL 2.98 | MAE 20.25m

--- VALIDATION: MSE 1164.99 | ANEES nan ---
----------------------------------------
Iter 190 (MSE_WARMUP): Loss 180.4816 | MSE 180.48 | NLL 2.70 | MAE 7.89m
Iter 200 (MSE_WARMUP): Loss 77.0311 | MSE 77.03 | NLL 2.58 | MAE 5.20m

--- VALIDATION: MSE 620.51 | ANEES nan ---
----------------------------------------
Iter 210 (MSE_WARMUP): Loss 601.4601 | MSE 601.46 | NLL 2.93 | MAE 10.78m
```

```
Iter 220 (MSE_WARMUP): Loss 137.3063 | MSE 137.30 | NLL 2.83 | MAE 6.99m

--- VALIDATION: MSE 1368.83 | ANEES nan ---
----------------------------------------
Iter 230 (MSE_WARMUP): Loss 93.6367 | MSE 93.64 | NLL 2.67 | MAE 5.81m
Iter 240 (MSE_WARMUP): Loss 5620.1108 | MSE 5620.11 | NLL 2.97 | MAE 18.73m

--- VALIDATION: MSE 445.94 | ANEES nan ---
----------------------------------------
Iter 250 (MSE_WARMUP): Loss 149.6625 | MSE 149.66 | NLL 2.61 | MAE 7.35m
Iter 260 (MSE_WARMUP): Loss 15469.8672 | MSE 15469.87 | NLL 2.85 | MAE 14.13m

--- VALIDATION: MSE 1268.36 | ANEES nan ---
----------------------------------------
Iter 270 (MSE_WARMUP): Loss 22482.1055 | MSE 22482.10 | NLL 2.81 | MAE 21.41m
Iter 280 (MSE_WARMUP): Loss 247.1103 | MSE 247.11 | NLL 2.84 | MAE 8.26m

--- VALIDATION: MSE 313.95 | ANEES nan ---
----------------------------------------
Iter 290 (MSE_WARMUP): Loss 75.8491 | MSE 75.85 | NLL 2.76 | MAE 5.25m
Iter 300 (MSE_WARMUP): Loss 231.7278 | MSE 231.73 | NLL 2.97 | MAE 9.01m

--- VALIDATION: MSE 620.76 | ANEES nan ---
----------------------------------------
Iter 310 (MSE_WARMUP): Loss 153.8993 | MSE 153.90 | NLL 2.81 | MAE 7.26m
Iter 320 (MSE_WARMUP): Loss 75.4159 | MSE 75.41 | NLL 2.71 | MAE 5.21m

--- VALIDATION: MSE 1177.80 | ANEES nan ---
----------------------------------------
Iter 330 (MSE_WARMUP): Loss 287.1146 | MSE 287.11 | NLL 2.78 | MAE 9.77m
Iter 340 (MSE_WARMUP): Loss 156.9136 | MSE 156.91 | NLL 2.77 | MAE 7.54m

--- VALIDATION: MSE 347.73 | ANEES nan ---
----------------------------------------
Iter 350 (MSE_WARMUP): Loss 60.4849 | MSE 60.48 | NLL 2.74 | MAE 4.79m
Iter 360 (MSE_WARMUP): Loss 177.1893 | MSE 177.19 | NLL 2.73 | MAE 8.04m

--- VALIDATION: MSE 1074.95 | ANEES nan ---
----------------------------------------
Iter 370 (MSE_WARMUP): Loss 162.9735 | MSE 162.97 | NLL 2.66 | MAE 7.59m
Iter 380 (MSE_WARMUP): Loss 315.1259 | MSE 315.12 | NLL 2.80 | MAE 5.60m

--- VALIDATION: MSE 1393.97 | ANEES nan ---
----------------------------------------
Iter 390 (MSE_WARMUP): Loss 2833.7051 | MSE 2833.70 | NLL 2.86 | MAE 11.76m
Iter 400 (MSE_WARMUP): Loss 1736.2002 | MSE 1736.20 | NLL 2.70 | MAE 10.71m

--- VALIDATION: MSE 437.00 | ANEES nan ---
```

```
----------------------------------------
Iter 410 (MSE_WARMUP): Loss 92.6200 | MSE 92.62 | NLL 2.76 | MAE 5.68m
Iter 420 (MSE_WARMUP): Loss 516.2682 | MSE 516.27 | NLL 2.80 | MAE 10.99m

--- VALIDATION: MSE 1384.42 | ANEES nan ---
----------------------------------------
Iter 430 (MSE_WARMUP): Loss 1834.8528 | MSE 1834.85 | NLL 2.78 | MAE 10.94m
Iter 440 (MSE_WARMUP): Loss 387.5937 | MSE 387.59 | NLL 2.70 | MAE 6.48m

--- VALIDATION: MSE 315.93 | ANEES nan ---
----------------------------------------
Iter 450 (MSE_WARMUP): Loss 1484.2990 | MSE 1484.30 | NLL 2.90 | MAE 11.71m
Iter 460 (MSE_WARMUP): Loss 182.1661 | MSE 182.16 | NLL 2.75 | MAE 7.65m

--- VALIDATION: MSE 392.54 | ANEES nan ---
----------------------------------------
Iter 470 (MSE_WARMUP): Loss 85.1451 | MSE 85.14 | NLL 2.63 | MAE 5.70m
Iter 480 (MSE_WARMUP): Loss 239.3535 | MSE 239.35 | NLL 2.74 | MAE 9.17m

--- VALIDATION: MSE 421.23 | ANEES nan ---
----------------------------------------
Iter 490 (MSE_WARMUP): Loss 144.2996 | MSE 144.30 | NLL 2.70 | MAE 7.16m
Iter 500 (MSE_WARMUP): Loss 80.1119 | MSE 80.11 | NLL 2.62 | MAE 5.40m

--- VALIDATION: MSE 299.04 | ANEES nan ---
----------------------------------------
Iter 510 (NLL_OPTIM): Loss 27787.4082 | MSE 277.84 | NLL 2.99 | MAE 9.84m
Iter 520 (NLL_OPTIM): Loss 19428.8027 | MSE 194.26 | NLL 2.74 | MAE 7.76m

--- VALIDATION: MSE 1166.03 | ANEES nan ---
----------------------------------------
Iter 530 (NLL_OPTIM): Loss 550688.0625 | MSE 5506.85 | NLL 2.83 | MAE 9.82m
Iter 540 (NLL_OPTIM): Loss 3168762.0000 | MSE 31687.59 | NLL 2.70 | MAE 19.28m

--- VALIDATION: MSE 422.14 | ANEES nan ---
----------------------------------------
Iter 550 (NLL_OPTIM): Loss 566919.5625 | MSE 5669.17 | NLL 2.69 | MAE 11.90m
Iter 560 (NLL_OPTIM): Loss 7479.7378 | MSE 74.77 | NLL 2.74 | MAE 5.25m

--- VALIDATION: MSE 688.83 | ANEES nan ---
----------------------------------------
Iter 570 (NLL_OPTIM): Loss 54862.0508 | MSE 548.59 | NLL 2.97 | MAE 10.21m
Iter 580 (NLL_OPTIM): Loss 14426.9062 | MSE 144.24 | NLL 2.75 | MAE 7.18m

--- VALIDATION: MSE 393.33 | ANEES nan ---
----------------------------------------
Iter 590 (NLL_OPTIM): Loss 19972.3516 | MSE 199.69 | NLL 2.98 | MAE 6.05m
Iter 600 (NLL_OPTIM): Loss 163486.9219 | MSE 1634.84 | NLL 2.92 | MAE 13.32m
```

```
--- VALIDATION: MSE 318.48 | ANEES nan ---
----------------------------------------
Iter 610 (NLL_OPTIM): Loss 18746.2812 | MSE 187.44 | NLL 2.69 | MAE 8.22m
Iter 620 (NLL_OPTIM): Loss 7505.6113 | MSE 75.03 | NLL 2.64 | MAE 5.11m

--- VALIDATION: MSE 857.86 | ANEES nan ---
----------------------------------------
Iter 630 (NLL_OPTIM): Loss 189253.4219 | MSE 1892.51 | NLL 2.91 | MAE 12.14m
Iter 640 (NLL_OPTIM): Loss 795555.8125 | MSE 7955.53 | NLL 2.68 | MAE 14.10m

--- VALIDATION: MSE 573.15 | ANEES nan ---
----------------------------------------
Iter 650 (NLL_OPTIM): Loss 7243.4521 | MSE 72.41 | NLL 2.77 | MAE 5.24m
Iter 660 (NLL_OPTIM): Loss 22178.0605 | MSE 221.75 | NLL 2.78 | MAE 8.93m

--- VALIDATION: MSE 537.52 | ANEES nan ---
----------------------------------------
Iter 670 (NLL_OPTIM): Loss 16901.5352 | MSE 168.99 | NLL 2.78 | MAE 7.96m
Iter 680 (NLL_OPTIM): Loss 7682.1030 | MSE 76.79 | NLL 2.66 | MAE 5.33m

--- VALIDATION: MSE 395.26 | ANEES nan ---
----------------------------------------
Iter 690 (NLL_OPTIM): Loss 22393.0254 | MSE 223.90 | NLL 2.81 | MAE 8.83m
Iter 700 (NLL_OPTIM): Loss 92882.7422 | MSE 928.80 | NLL 2.65 | MAE 9.04m

--- VALIDATION: MSE 1047.96 | ANEES nan ---
----------------------------------------
Iter 710 (NLL_OPTIM): Loss 7117.3989 | MSE 71.14 | NLL 3.11 | MAE 4.98m
Iter 720 (NLL_OPTIM): Loss 39043.5898 | MSE 390.41 | NLL 2.80 | MAE 11.00m

--- VALIDATION: MSE 590.48 | ANEES nan ---
----------------------------------------
Iter 730 (NLL_OPTIM): Loss 3318241.5000 | MSE 33182.39 | NLL 2.86 | MAE 22.12m
Iter 740 (NLL_OPTIM): Loss 6958.6719 | MSE 69.56 | NLL 2.76 | MAE 5.02m

--- VALIDATION: MSE 2497.74 | ANEES nan ---
----------------------------------------
Iter 750 (NLL_OPTIM): Loss 142248.2031 | MSE 1422.45 | NLL 2.83 | MAE 11.58m
Iter 760 (NLL_OPTIM): Loss 16379.3525 | MSE 163.77 | NLL 2.79 | MAE 7.79m

--- VALIDATION: MSE 801.91 | ANEES nan ---
----------------------------------------
Iter 770 (NLL_OPTIM): Loss 43162.5117 | MSE 431.60 | NLL 2.76 | MAE 6.72m
Iter 780 (NLL_OPTIM): Loss 199387.5469 | MSE 1993.85 | NLL 2.97 | MAE 11.87m

--- VALIDATION: MSE 1140.12 | ANEES nan ---
----------------------------------------
```

```
Iter 790 (NLL_OPTIM): Loss 124404.6641 | MSE 1244.02 | NLL 2.80 | MAE 9.68m
Iter 800 (NLL_OPTIM): Loss 8167.3120 | MSE 81.64 | NLL 2.83 | MAE 5.40m

--- VALIDATION: MSE 1080.74 | ANEES nan ---
----------------------------------------
Iter 810 (NLL_OPTIM): Loss 22828.7148 | MSE 228.26 | NLL 2.86 | MAE 9.29m
Iter 820 (NLL_OPTIM): Loss 16332.5361 | MSE 163.30 | NLL 2.76 | MAE 7.55m

--- VALIDATION: MSE 1122.70 | ANEES nan ---
----------------------------------------
Iter 830 (NLL_OPTIM): Loss 8747.2939 | MSE 87.45 | NLL 2.74 | MAE 5.59m
Iter 840 (NLL_OPTIM): Loss 24979.9355 | MSE 249.77 | NLL 2.82 | MAE 9.48m

--- VALIDATION: MSE 1323.57 | ANEES nan ---
----------------------------------------
Iter 850 (NLL_OPTIM): Loss 15082.3389 | MSE 150.80 | NLL 2.59 | MAE 7.38m
Iter 860 (NLL_OPTIM): Loss 7206.1294 | MSE 72.03 | NLL 2.64 | MAE 5.11m

--- VALIDATION: MSE 1336.63 | ANEES nan ---
----------------------------------------
Iter 870 (NLL_OPTIM): Loss 144329.1406 | MSE 1443.26 | NLL 3.11 | MAE 11.75m
Iter 880 (NLL_OPTIM): Loss 13841.4297 | MSE 138.39 | NLL 2.63 | MAE 7.04m

--- VALIDATION: MSE 746.84 | ANEES nan ---
----------------------------------------
Iter 890 (NLL_OPTIM): Loss 324059.6562 | MSE 3240.57 | NLL 2.68 | MAE 8.03m
Iter 900 (NLL_OPTIM): Loss 91814.2344 | MSE 918.12 | NLL 2.71 | MAE 10.56m

--- VALIDATION: MSE 392.73 | ANEES nan ---
----------------------------------------
Iter 910 (NLL_OPTIM): Loss 14633.8018 | MSE 146.31 | NLL 2.76 | MAE 7.55m
Iter 920 (NLL_OPTIM): Loss 6492.4756 | MSE 64.90 | NLL 2.75 | MAE 5.03m

--- VALIDATION: MSE 1200.93 | ANEES nan ---
----------------------------------------
Iter 930 (NLL_OPTIM): Loss 21057.6465 | MSE 210.55 | NLL 3.00 | MAE 8.37m
Iter 940 (NLL_OPTIM): Loss 14500.4082 | MSE 144.98 | NLL 2.69 | MAE 7.15m

--- VALIDATION: MSE 1805.03 | ANEES nan ---
----------------------------------------
Iter 950 (NLL_OPTIM): Loss 7351.3384 | MSE 73.49 | NLL 2.60 | MAE 5.23m
Iter 960 (NLL_OPTIM): Loss 21310.0801 | MSE 213.07 | NLL 2.76 | MAE 8.38m

--- VALIDATION: MSE 669.76 | ANEES nan ---
----------------------------------------
Iter 970 (NLL_OPTIM): Loss 17426.2734 | MSE 174.24 | NLL 2.70 | MAE 7.84m
Iter 980 (NLL_OPTIM): Loss 7243.2681 | MSE 72.41 | NLL 2.74 | MAE 5.27m
```

```
--- VALIDATION: MSE 356.73 | ANEES nan ---
----------------------------------------
Iter 990 (NLL_OPTIM): Loss 24296.6328 | MSE 242.94 | NLL 2.89 | MAE 9.27m
Iter 1000 (NLL_OPTIM): Loss 1363233.1250 | MSE 13632.30 | NLL 2.82 | MAE 14.03m

--- VALIDATION: MSE 243.63 | ANEES nan ---
  >>> New Best Model! (MSE: 243.63) <<<
----------------------------------------
Iter 1010 (NLL_OPTIM): Loss 5944.1494 | MSE 59.41 | NLL 2.71 | MAE 4.77m
Iter 1020 (NLL_OPTIM): Loss 18616.0312 | MSE 186.13 | NLL 2.85 | MAE 7.97m

--- VALIDATION: MSE 675.78 | ANEES nan ---
----------------------------------------
Iter 1030 (NLL_OPTIM): Loss 14398.5430 | MSE 143.96 | NLL 2.75 | MAE 7.21m
Iter 1040 (NLL_OPTIM): Loss 6983.3784 | MSE 69.81 | NLL 2.59 | MAE 5.17m

--- VALIDATION: MSE 362.99 | ANEES nan ---
----------------------------------------
Iter 1050 (NLL_OPTIM): Loss 68971.7656 | MSE 689.69 | NLL 2.78 | MAE 10.64m
Iter 1060 (NLL_OPTIM): Loss 43917.2695 | MSE 439.15 | NLL 2.70 | MAE 8.88m

--- VALIDATION: MSE 1619.13 | ANEES nan ---
----------------------------------------
Iter 1070 (NLL_OPTIM): Loss 568918.2500 | MSE 5689.16 | NLL 2.68 | MAE 10.61m
Iter 1080 (NLL_OPTIM): Loss 3174456.0000 | MSE 31744.53 | NLL 2.93 | MAE 19.77m

--- VALIDATION: MSE 455.21 | ANEES nan ---
----------------------------------------
Iter 1090 (NLL_OPTIM): Loss 15424.0732 | MSE 154.21 | NLL 2.77 | MAE 7.55m
Iter 1100 (NLL_OPTIM): Loss 31364.8027 | MSE 313.62 | NLL 2.49 | MAE 5.64m

--- VALIDATION: MSE 585.63 | ANEES nan ---
----------------------------------------
Iter 1110 (NLL_OPTIM): Loss 396763.4062 | MSE 3967.61 | NLL 2.85 | MAE 13.77m
Iter 1120 (NLL_OPTIM): Loss 16498.1328 | MSE 164.95 | NLL 2.76 | MAE 7.63m

--- VALIDATION: MSE 245.73 | ANEES nan ---
----------------------------------------
Iter 1130 (NLL_OPTIM): Loss 138867.6094 | MSE 1388.65 | NLL 2.80 | MAE 7.32m
Iter 1140 (NLL_OPTIM): Loss 1291385.1250 | MSE 12913.82 | NLL 2.91 | MAE 16.03m

--- VALIDATION: MSE 191.92 | ANEES nan ---
  >>> New Best Model! (MSE: 191.92) <<<
----------------------------------------
Iter 1150 (NLL_OPTIM): Loss 15153.0410 | MSE 151.50 | NLL 2.84 | MAE 7.32m
Iter 1160 (NLL_OPTIM): Loss 9121.9580 | MSE 91.19 | NLL 2.66 | MAE 5.63m

--- VALIDATION: MSE 305.48 | ANEES nan ---
```

```
----------------------------------------
Iter 1170 (NLL_OPTIM): Loss 264578.5938 | MSE 2645.76 | NLL 2.96 | MAE 12.22m
Iter 1180 (NLL_OPTIM): Loss 14452.9043 | MSE 144.50 | NLL 2.92 | MAE 7.27m

--- VALIDATION: MSE 3702.62 | ANEES nan ---
----------------------------------------
Iter 1190 (NLL_OPTIM): Loss 8073.2822 | MSE 80.71 | NLL 2.63 | MAE 5.57m
Iter 1200 (NLL_OPTIM): Loss 24603.5430 | MSE 246.01 | NLL 3.03 | MAE 9.35m

--- VALIDATION: MSE 629.38 | ANEES nan ---
----------------------------------------
Iter 1210 (NLL_OPTIM): Loss 87829.8125 | MSE 878.27 | NLL 2.84 | MAE 9.04m
Iter 1220 (NLL_OPTIM): Loss 31952.3105 | MSE 319.50 | NLL 2.72 | MAE 5.99m

--- VALIDATION: MSE 815.77 | ANEES nan ---
----------------------------------------
Iter 1230 (NLL_OPTIM): Loss 348077.5938 | MSE 3480.75 | NLL 2.96 | MAE 13.79m
Iter 1240 (NLL_OPTIM): Loss 1955430.3750 | MSE 19554.28 | NLL 2.68 | MAE 16.32m

--- VALIDATION: MSE 355.83 | ANEES nan ---
----------------------------------------
Iter 1250 (NLL_OPTIM): Loss 6712.0073 | MSE 67.09 | NLL 2.98 | MAE 4.97m
Iter 1260 (NLL_OPTIM): Loss 174261.1406 | MSE 1742.58 | NLL 2.94 | MAE 12.40m

--- VALIDATION: MSE 1217.13 | ANEES nan ---
----------------------------------------
Iter 1270 (NLL_OPTIM): Loss 55514.6797 | MSE 555.12 | NLL 2.74 | MAE 9.54m
Iter 1280 (NLL_OPTIM): Loss 7340.5977 | MSE 73.38 | NLL 2.77 | MAE 5.19m

--- VALIDATION: MSE 725.43 | ANEES nan ---
----------------------------------------
Iter 1290 (NLL_OPTIM): Loss 24870.1895 | MSE 248.67 | NLL 2.82 | MAE 8.72m
Iter 1300 (NLL_OPTIM): Loss 16916.8633 | MSE 169.14 | NLL 2.71 | MAE 7.80m

--- VALIDATION: MSE 1153.99 | ANEES nan ---
----------------------------------------
Iter 1310 (NLL_OPTIM): Loss 8343.8145 | MSE 83.41 | NLL 2.67 | MAE 5.48m
Iter 1320 (NLL_OPTIM): Loss 24244.0742 | MSE 242.41 | NLL 2.84 | MAE 9.17m

--- VALIDATION: MSE 732.32 | ANEES nan ---
----------------------------------------
Iter 1330 (NLL_OPTIM): Loss 556850.0000 | MSE 5568.47 | NLL 2.84 | MAE 11.58m
Iter 1340 (NLL_OPTIM): Loss 7906.6704 | MSE 79.04 | NLL 2.84 | MAE 5.51m

--- VALIDATION: MSE 529.49 | ANEES nan ---
----------------------------------------
Iter 1350 (NLL_OPTIM): Loss 28894.8066 | MSE 288.92 | NLL 2.80 | MAE 10.10m
Iter 1360 (NLL_OPTIM): Loss 15201.5654 | MSE 151.99 | NLL 2.89 | MAE 7.56m
```

```
--- VALIDATION: MSE 3977.11 | ANEES nan ---
----------------------------------------
Iter 1370 (NLL_OPTIM): Loss 20291.1699 | MSE 202.89 | NLL 2.62 | MAE 5.92m
Iter 1380 (NLL_OPTIM): Loss 266240.5312 | MSE 2662.38 | NLL 2.87 | MAE 13.59m

--- VALIDATION: MSE 479.50 | ANEES nan ---
----------------------------------------
Iter 1390 (NLL_OPTIM): Loss 15909.7441 | MSE 159.07 | NLL 2.68 | MAE 7.27m
Iter 1400 (NLL_OPTIM): Loss 313840.6562 | MSE 3138.38 | NLL 2.79 | MAE 8.45m

--- VALIDATION: MSE 488.01 | ANEES nan ---
----------------------------------------
Iter 1410 (NLL_OPTIM): Loss 1763777.0000 | MSE 17637.74 | NLL 2.79 | MAE 15.39m
Iter 1420 (NLL_OPTIM): Loss 16159.0742 | MSE 161.56 | NLL 2.81 | MAE 7.64m

--- VALIDATION: MSE 1294.95 | ANEES nan ---
----------------------------------------
Iter 1430 (NLL_OPTIM): Loss 7787.2168 | MSE 77.85 | NLL 2.67 | MAE 5.36m
Iter 1440 (NLL_OPTIM): Loss 34707.3750 | MSE 347.04 | NLL 2.96 | MAE 10.10m

--- VALIDATION: MSE 630.45 | ANEES nan ---
----------------------------------------
Iter 1450 (NLL_OPTIM): Loss 125040.0938 | MSE 1250.37 | NLL 2.64 | MAE 9.25m
Iter 1460 (NLL_OPTIM): Loss 9717.1270 | MSE 97.14 | NLL 2.78 | MAE 5.70m

--- VALIDATION: MSE 535.64 | ANEES nan ---
----------------------------------------
Iter 1470 (NLL_OPTIM): Loss 235735.0312 | MSE 2357.32 | NLL 2.90 | MAE 11.34m
Iter 1480 (NLL_OPTIM): Loss 1353039.7500 | MSE 13530.37 | NLL 2.92 | MAE 14.03m

--- VALIDATION: MSE 252.26 | ANEES nan ---
----------------------------------------
Iter 1490 (NLL_OPTIM): Loss 26099.1621 | MSE 260.97 | NLL 2.62 | MAE 5.75m
Iter 1500 (NLL_OPTIM): Loss 125628.8672 | MSE 1256.26 | NLL 2.85 | MAE 11.82m

--- VALIDATION: MSE 288.95 | ANEES nan ---
----------------------------------------
Iter 1510 (NLL_OPTIM): Loss 13373.0947 | MSE 133.70 | NLL 2.72 | MAE 6.84m
Iter 1520 (NLL_OPTIM): Loss 27777.4004 | MSE 277.75 | NLL 2.89 | MAE 5.95m

--- VALIDATION: MSE 1669.54 | ANEES nan ---
----------------------------------------
Iter 1530 (NLL_OPTIM): Loss 268274.2812 | MSE 2682.71 | NLL 2.89 | MAE 12.91m
Iter 1540 (NLL_OPTIM): Loss 112632.5938 | MSE 1126.30 | NLL 2.73 | MAE 10.06m

--- VALIDATION: MSE 445.96 | ANEES nan ---
----------------------------------------
```

```
Iter 1550 (NLL_OPTIM): Loss 850850.2500 | MSE 8508.48 | NLL 2.65 | MAE 11.37m
Iter 1560 (NLL_OPTIM): Loss 4884543.5000 | MSE 48845.41 | NLL 3.00 | MAE 22.69m

--- VALIDATION: MSE 539.13 | ANEES nan ---
----------------------------------------
Iter 1570 (NLL_OPTIM): Loss 45138.5000 | MSE 451.36 | NLL 2.89 | MAE 8.23m
Iter 1580 (NLL_OPTIM): Loss 39848.4883 | MSE 398.46 | NLL 2.81 | MAE 6.18m

--- VALIDATION: MSE 242.34 | ANEES nan ---
----------------------------------------
Iter 1590 (NLL_OPTIM): Loss 248858.5781 | MSE 2488.56 | NLL 2.93 | MAE 11.45m
Iter 1600 (NLL_OPTIM): Loss 84264.4609 | MSE 842.62 | NLL 2.69 | MAE 8.45m

--- VALIDATION: MSE 872.39 | ANEES nan ---
----------------------------------------
Iter 1610 (NLL_OPTIM): Loss 7321.7759 | MSE 73.19 | NLL 2.65 | MAE 5.17m
Iter 1620 (NLL_OPTIM): Loss 700291.7500 | MSE 7002.89 | NLL 2.80 | MAE 16.18m

--- VALIDATION: MSE 186.95 | ANEES nan ---
  >>> New Best Model! (MSE: 186.95) <<<
----------------------------------------
Iter 1630 (NLL_OPTIM): Loss 16580.7773 | MSE 165.78 | NLL 2.72 | MAE 7.61m
Iter 1640 (NLL_OPTIM): Loss 7952.3657 | MSE 79.50 | NLL 2.59 | MAE 5.51m

--- VALIDATION: MSE 201.30 | ANEES nan ---
----------------------------------------
Iter 1650 (NLL_OPTIM): Loss 101263.6953 | MSE 1012.61 | NLL 3.00 | MAE 11.81m
Iter 1660 (NLL_OPTIM): Loss 14747.3604 | MSE 147.44 | NLL 2.92 | MAE 7.18m

--- VALIDATION: MSE 215.81 | ANEES nan ---
----------------------------------------
Iter 1670 (NLL_OPTIM): Loss 536084.5625 | MSE 5360.81 | NLL 3.17 | MAE 10.97m
Iter 1680 (NLL_OPTIM): Loss 3905300.5000 | MSE 39052.98 | NLL 2.80 | MAE 37.08m

--- VALIDATION: MSE 159.29 | ANEES nan ---
  >>> New Best Model! (MSE: 159.29) <<<
----------------------------------------
Iter 1690 (NLL_OPTIM): Loss 16273.0381 | MSE 162.70 | NLL 2.79 | MAE 7.61m
Iter 1700 (NLL_OPTIM): Loss 7585.1729 | MSE 75.82 | NLL 2.72 | MAE 5.27m

--- VALIDATION: MSE 667.63 | ANEES nan ---
----------------------------------------
Iter 1710 (NLL_OPTIM): Loss 20689.0996 | MSE 206.86 | NLL 2.99 | MAE 8.52m
Iter 1720 (NLL_OPTIM): Loss 157608.9375 | MSE 1576.06 | NLL 2.78 | MAE 10.51m

--- VALIDATION: MSE 568.42 | ANEES nan ---
----------------------------------------
Iter 1730 (NLL_OPTIM): Loss 7706.4023 | MSE 77.04 | NLL 2.87 | MAE 5.36m
```

```
Iter 1740 (NLL_OPTIM): Loss 22523.4180 | MSE 225.20 | NLL 2.94 | MAE 8.60m

--- VALIDATION: MSE 2380.57 | ANEES nan ---
----------------------------------------
Iter 1750 (NLL_OPTIM): Loss 615497.0000 | MSE 6154.94 | NLL 2.75 | MAE 11.90m
Iter 1760 (NLL_OPTIM): Loss 17568.2402 | MSE 175.65 | NLL 2.80 | MAE 5.50m

--- VALIDATION: MSE 625.26 | ANEES nan ---
----------------------------------------
Iter 1770 (NLL_OPTIM): Loss 233762.6875 | MSE 2337.60 | NLL 2.83 | MAE 13.16m
Iter 1780 (NLL_OPTIM): Loss 14791.8213 | MSE 147.89 | NLL 2.83 | MAE 7.31m

--- VALIDATION: MSE 2145.25 | ANEES nan ---
----------------------------------------
Iter 1790 (NLL_OPTIM): Loss 6727.4814 | MSE 67.25 | NLL 2.53 | MAE 5.02m
Iter 1800 (NLL_OPTIM): Loss 350518.5000 | MSE 3505.16 | NLL 2.93 | MAE 13.27m

--- VALIDATION: MSE 3022.92 | ANEES nan ---
----------------------------------------
Iter 1810 (NLL_OPTIM): Loss 17150.3164 | MSE 171.47 | NLL 2.90 | MAE 7.57m
Iter 1820 (NLL_OPTIM): Loss 8602.1963 | MSE 86.00 | NLL 2.67 | MAE 5.52m

--- VALIDATION: MSE 609.00 | ANEES nan ---
----------------------------------------
Iter 1830 (NLL_OPTIM): Loss 25957.6777 | MSE 259.55 | NLL 2.81 | MAE 9.32m
Iter 1840 (NLL_OPTIM): Loss 15775.4434 | MSE 157.73 | NLL 2.67 | MAE 7.39m

--- VALIDATION: MSE 1070.99 | ANEES nan ---
----------------------------------------
Iter 1850 (NLL_OPTIM): Loss 5792.0737 | MSE 57.89 | NLL 2.71 | MAE 4.80m
Iter 1860 (NLL_OPTIM): Loss 21666.7871 | MSE 216.64 | NLL 2.79 | MAE 8.56m

--- VALIDATION: MSE 1244.84 | ANEES nan ---
----------------------------------------
Iter 1870 (NLL_OPTIM): Loss 13856.7119 | MSE 138.54 | NLL 2.70 | MAE 6.94m
Iter 1880 (NLL_OPTIM): Loss 7435.9497 | MSE 74.33 | NLL 2.65 | MAE 5.05m

--- VALIDATION: MSE 1071.44 | ANEES nan ---
----------------------------------------
Iter 1890 (NLL_OPTIM): Loss 187832.7812 | MSE 1878.30 | NLL 2.91 | MAE 12.42m
Iter 1900 (NLL_OPTIM): Loss 52938.5781 | MSE 529.36 | NLL 2.88 | MAE 8.56m

--- VALIDATION: MSE 374.88 | ANEES nan ---
----------------------------------------
Iter 1910 (NLL_OPTIM): Loss 1370780.3750 | MSE 13707.78 | NLL 2.85 | MAE 14.27m
Iter 1920 (NLL_OPTIM): Loss 7795950.5000 | MSE 77959.48 | NLL 2.95 | MAE 26.79m

--- VALIDATION: MSE 1124.50 | ANEES nan ---
```

```
----------------------------------------
Iter 1930 (NLL_OPTIM): Loss 14892.9863 | MSE 148.90 | NLL 2.62 | MAE 7.35m
Iter 1940 (NLL_OPTIM): Loss 7152.2915 | MSE 71.50 | NLL 2.72 | MAE 5.25m

--- VALIDATION: MSE 414.04 | ANEES nan ---
----------------------------------------
Iter 1950 (NLL_OPTIM): Loss 92398.7812 | MSE 923.96 | NLL 2.98 | MAE 10.68m
Iter 1960 (NLL_OPTIM): Loss 15350.6943 | MSE 153.48 | NLL 2.67 | MAE 7.35m

--- VALIDATION: MSE 1123.64 | ANEES nan ---
----------------------------------------
Iter 1970 (NLL_OPTIM): Loss 35311.1328 | MSE 353.08 | NLL 2.80 | MAE 6.05m
Iter 1980 (NLL_OPTIM): Loss 233768.1250 | MSE 2337.65 | NLL 2.81 | MAE 12.69m

--- VALIDATION: MSE 1350.40 | ANEES nan ---
----------------------------------------
Iter 1990 (NLL_OPTIM): Loss 16357.7783 | MSE 163.55 | NLL 2.77 | MAE 7.73m
Iter 2000 (NLL_OPTIM): Loss 7148.9551 | MSE 71.46 | NLL 2.52 | MAE 4.98m

--- VALIDATION: MSE 839.25 | ANEES nan ---
----------------------------------------
Training completed.
 Fáze 3 dokončena. Model uložen do: bkn_curriculum_phase3_len300.pth

 Celý trénink dokončen.
```

[16]:
```python
if False:
    # save model.
    save_path = f'most_consistent_and_accurate_bknet.pth'
    torch.save(state_knet2.state_dict(), save_path)
    print(f"Model saved to '{save_path}'.")
```

```
Model saved to 'most_consistent_and_accurate_bknet.pth'.
```

1 Test na syntetické trajektorii

[17]:
```python
import torch
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import Filters
import os
from tqdm import tqdm
from Filters import TAN

# === KONFIGURACE ===
TEST_DATA_PATH = './generated_data_synthetic_controlled/test_set/test.pt'
```

```
PLOT_PER_ITERATION = True  # Vykreslovat graf pro každou trajektorii?
MAX_TEST_SAMPLES = 20      # Kolik trajektorií z test setu vyhodnotit
J_EVALUATION = 100         # Počet Monte Carlo vzorků pro BKN (Ensemble size)

print(f"=== VYHODNOCENÍ BKN NA TESTOVACÍ SADĚ (s ANEES) ===")
print(f"Načítám data z: {TEST_DATA_PATH}")

# 1. Načtení Testovací sady
if not os.path.exists(TEST_DATA_PATH):
    raise FileNotFoundError(f"Soubor {TEST_DATA_PATH} neexistuje!")

# Předpokládáme, že 'device' je definováno
if 'device' not in globals():
    device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

test_data = torch.load(TEST_DATA_PATH, map_location=device)
X_test_all = test_data['x']  # Ground Truth [N, Seq, 4]
Y_test_all = test_data['y']  # Measurements [N, Seq, 3]

n_samples = min(X_test_all.shape[0], MAX_TEST_SAMPLES)
print(f"Počet testovacích trajektorií: {n_samples}")
print(f"Ensemble size (BKN): {J_EVALUATION}")
print(f"Délka sekvence: {X_test_all.shape[1]}")
print("Modely: BKN vs. UKF vs. PF vs. APF")

# 2. Inicializace pro sběr dat
detailed_results = []
agg_mse = {"BKN": [], "UKF": [], "PF": [], "APF": []}
agg_pos = {"BKN": [], "UKF": [], "PF": [], "APF": []}
agg_anees = {"BKN": [], "UKF": [], "PF": [], "APF": []} # Nový list pro ANEES

# Ujistíme se, že BKN je v eval módu
state_knet2.eval()

# --- POMOCNÁ FUNKCE PRO ANEES ---
def calculate_anees(gt, est, P):
    """
    Vypočítá Average Normalized Estimation Error Squared.
    gt: Ground Truth [T, Dim] (NumPy)
    est: Odhad [T, Dim] (NumPy)
    P: Kovariační matice [T, Dim, Dim] (NumPy)
    """
    T = min(len(gt), len(est), len(P))
    anees_vals = []

    # Oříznutí na stejnou délku
    gt = gt[:T]
```

```
    est = est[:T]
    P = P[:T]

    for t in range(T):
        e_t = gt[t] - est[t] # Chyba v čase t
        P_t = P[t]

        try:
            # Inverze kovariance
            # Přičteme malé epsilon na diagonálu pro numerickou stabilitu,␣
 ↪pokud je singulární
            if np.linalg.cond(P_t) > 1e10:
                P_t = P_t + np.eye(P_t.shape[0]) * 1e-6

            P_inv = np.linalg.inv(P_t)

            # Mahalanobisova vzdálenost: e^T * P^-1 * e
            anees_t = e_t.T @ P_inv @ e_t
            anees_vals.append(anees_t)
        except np.linalg.LinAlgError:
            anees_vals.append(np.nan)

    return np.nanmean(anees_vals)

# --- HLAVNÍ SMYČKA ---
for i in tqdm(range(n_samples), desc="Evaluace"):

    # A) Příprava dat
    x_gt_tensor = X_test_all[i].to(device)
    y_obs_tensor = Y_test_all[i].to(device)

    x_gt = x_gt_tensor.cpu().numpy()
    seq_len = x_gt.shape[0]
    true_init_state = x_gt_tensor[0]

    # --- B) BKN (Ensemble) ---
    with torch.no_grad():
        init_batch = true_init_state.unsqueeze(0).repeat(J_EVALUATION, 1)
        state_knet2.reset(batch_size=J_EVALUATION, initial_state=init_batch)

        bkn_preds = []
        y_input_batch = y_obs_tensor.unsqueeze(0).repeat(J_EVALUATION, 1, 1)

        for t in range(1, seq_len):
            y_t = y_input_batch[:, t, :]
            x_est, _ = state_knet2.step(y_t)
            bkn_preds.append(x_est)
```

```
        if len(bkn_preds) > 0:
            bkn_preds_tensor = torch.stack(bkn_preds, dim=1) # [J, Seq-1, 4]
            full_bkn_ensemble = torch.cat([init_batch.unsqueeze(1),␣
↪bkn_preds_tensor], dim=1) # [J, Seq, 4]

            # Mean Estimate
            x_est_mean = full_bkn_ensemble.mean(dim=0)
            x_est_bkn = x_est_mean.cpu().numpy()

            # --- VÝPOČET KOVARIANCE PRO BKN ---
            # P = 1/(J-1) * sum((x_j - x_mean) * (x_j - x_mean)^T)
            # Vycentrování
            residuals = full_bkn_ensemble - x_est_mean.unsqueeze(0) # [J, Seq,␣
↪4]
            # Permute pro batch matmul: [Seq, J, 4] a [Seq, 4, J]
            residuals = residuals.permute(1, 2, 0) # [Seq, 4, J]

            # Batch matrix multiplication: (Seq, 4, J) @ (Seq, J, 4) -> (Seq,␣
↪4, 4)
            P_bkn_tensor = torch.bmm(residuals, residuals.transpose(1, 2)) /␣
↪(J_EVALUATION - 1)
            # Přičtení process noise/stabilitu (volitelné, BKN variance je␣
↪epistemická)
            P_bkn = P_bkn_tensor.cpu().numpy()

        else:
            x_est_bkn = x_gt
            P_bkn = np.eye(4)[np.newaxis, :, :].repeat(len(x_gt), axis=0)

    # --- C) Klasické Filtry ---

    # UKF
    ukf_ideal = TAN.UnscentedKalmanFilterTAN(system_model)
    ukf_res = ukf_ideal.process_sequence(y_seq=y_obs_tensor,␣
↪Ex0=true_init_state, P0=system_model.P0)
    x_est_ukf = ukf_res['x_filtered'].cpu().numpy()
    # Získání P pro UKF (zkusíme různé klíče)
    P_ukf = ukf_res.get('P_filtered', ukf_res.get('P', None))
    if P_ukf is not None: P_ukf = P_ukf.cpu().numpy()

    # PF
    pf = TAN.ParticleFilterTAN(system_model, num_particles=1000)
    pf_res = pf.process_sequence(y_seq=y_obs_tensor, Ex0=true_init_state,␣
↪P0=system_model.P0)
    x_est_pf = pf_res['x_filtered'].cpu().numpy()
```

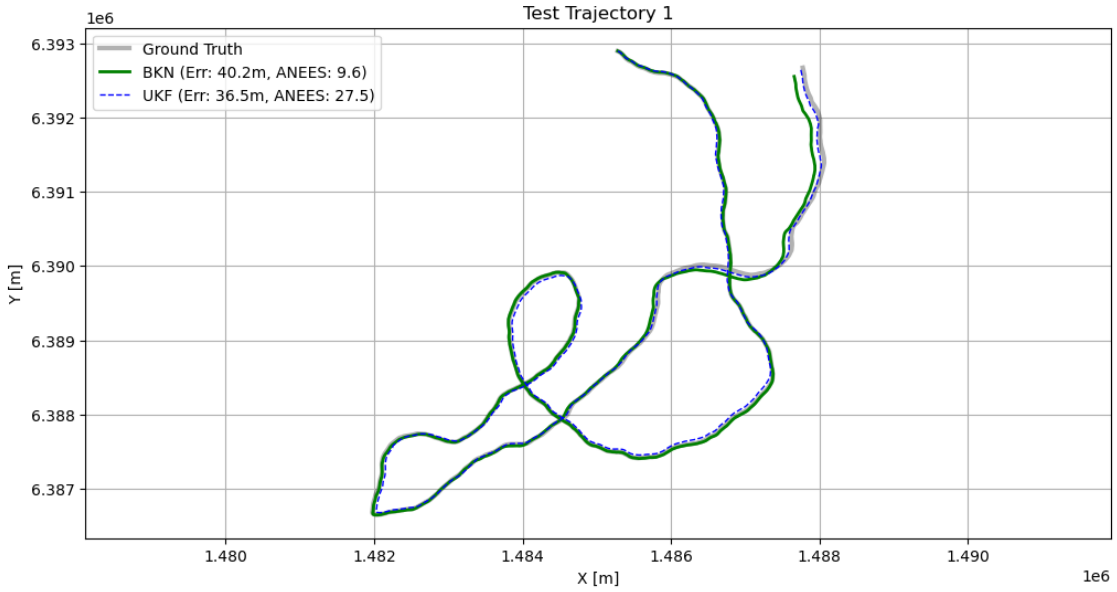
```
    P_pf = pf_res.get('P_filtered', pf_res.get('P', None))
    if P_pf is not None: P_pf = P_pf.cpu().numpy()

    # APF
    apf = TAN.AuxiliaryParticleFilterTAN(system_model, num_particles=2000)
    apf_res = apf.process_sequence(y_seq=y_obs_tensor, Ex0=true_init_state,␣
↪P0=system_model.P0)
    x_est_apf = apf_res['x_filtered'].cpu().numpy()
    P_apf = apf_res.get('P_filtered', apf_res.get('P', None))
    if P_apf is not None: P_apf = P_apf.cpu().numpy()

    # --- D) Výpočet chyb a ANEES ---
    min_len = min(len(x_gt), len(x_est_bkn), len(x_est_ukf))

    def calc_metrics(est, gt, P_mat):
        diff = est[:min_len] - gt[:min_len]
        mse = np.mean(np.sum(diff[:, :2]**2, axis=1))
        pos_err = np.mean(np.sqrt(diff[:, 0]**2 + diff[:, 1]**2))

        anees = np.nan
        if P_mat is not None:
            anees = calculate_anees(gt[:min_len], est[:min_len], P_mat[:
↪min_len])

        return mse, pos_err, anees

    # Calculate for all
    mse_bkn, pos_bkn, anees_bkn = calc_metrics(x_est_bkn, x_gt, P_bkn)
    mse_ukf, pos_ukf, anees_ukf = calc_metrics(x_est_ukf, x_gt, P_ukf)
    mse_pf, pos_pf, anees_pf = calc_metrics(x_est_pf, x_gt, P_pf)
    mse_apf, pos_apf, anees_apf = calc_metrics(x_est_apf, x_gt, P_apf)

    # Uložení
    agg_mse["BKN"].append(mse_bkn); agg_pos["BKN"].append(pos_bkn);␣
↪agg_anees["BKN"].append(anees_bkn)
    agg_mse["UKF"].append(mse_ukf); agg_pos["UKF"].append(pos_ukf);␣
↪agg_anees["UKF"].append(anees_ukf)
    agg_mse["PF"].append(mse_pf);   agg_pos["PF"].append(pos_pf);   ␣
↪agg_anees["PF"].append(anees_pf)
    agg_mse["APF"].append(mse_apf); agg_pos["APF"].append(pos_apf);␣
↪agg_anees["APF"].append(anees_apf)

    detailed_results.append({
        "Run_ID": i + 1,
        "BKN_PosErr": pos_bkn, "BKN_ANEES": anees_bkn,
        "UKF_PosErr": pos_ukf, "UKF_ANEES": anees_ukf,
```

```
        "PF_PosErr": pos_pf,   "PF_ANEES": anees_pf,
        "APF_PosErr": pos_apf, "APF_ANEES": anees_apf
    })

    # E) Vykreslení
    if PLOT_PER_ITERATION:
        fig = plt.figure(figsize=(12, 6))
        plt.plot(x_gt[:, 0], x_gt[:, 1], 'k-', linewidth=3, alpha=0.3,␣
 ↪label='Ground Truth')
        plt.plot(x_est_bkn[:, 0], x_est_bkn[:, 1], 'g-', linewidth=2,␣
 ↪label=f'BKN (Err: {pos_bkn:.1f}m, ANEES: {anees_bkn:.1f})')
        plt.plot(x_est_ukf[:, 0], x_est_ukf[:, 1], 'b--', linewidth=1,␣
 ↪label=f'UKF (Err: {pos_ukf:.1f}m, ANEES: {anees_ukf:.1f})')
        # Pro přehlednost vykreslíme jen BKN a UKF, případně odkomentujte PF/APF
        # plt.plot(x_est_pf[:, 0], x_est_pf[:, 1], 'r:', linewidth=1, alpha=0.
 ↪6, label='PF')
        plt.title(f"Test Trajectory {i+1}")
        plt.xlabel("X [m]")
        plt.ylabel("Y [m]")
        plt.legend()
        plt.axis('equal')
        plt.grid(True)
        plt.show()

# --- VÝPIS VÝSLEDKŮ ---
df_results = pd.DataFrame(detailed_results)
print("\n" + "="*120)
print(f"DETAILNÍ VÝSLEDKY (Pozice v metrech | ANEES - ideál ~4.0)")
print("="*120)
pd.options.display.float_format = '{:,.2f}'.format
print(df_results[["Run_ID", "BKN_PosErr", "BKN_ANEES", "UKF_PosErr",␣
 ↪"UKF_ANEES", "PF_PosErr", "APF_PosErr"]])

print("\n" + "="*120)
print(f"SOUHRNNÁ STATISTIKA ({n_samples} trajektorií)")
print("="*120)

def get_stats(key):
    return (np.nanmean(agg_mse[key]), np.nanstd(agg_mse[key]),
            np.nanmean(agg_pos[key]), np.nanstd(agg_pos[key]),
            np.nanmean(agg_anees[key]), np.nanstd(agg_anees[key]))

bkn_s = get_stats("BKN")
ukf_s = get_stats("UKF")
pf_s = get_stats("PF")
apf_s = get_stats("APF")
```

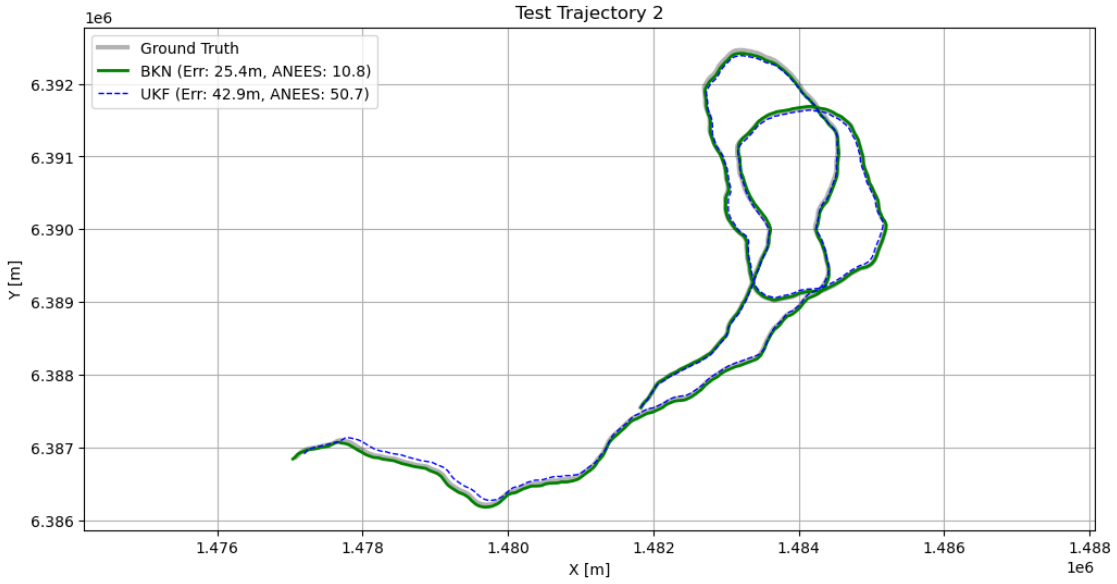
```
# Formátování tabulky
header = f"{'Model':<10} | {'Pos Error [m] (Mean ± Std)':<30} | {'ANEES (Mean ±␣
 ↪Std)':<30}"
print(header)
print("-" * len(header))
print(f"{'BKN':<10} | {bkn_s[2]:.2f} ± {bkn_s[3]:.2f} m {'':<14} | {bkn_s[4]:.
 ↪2f} ± {bkn_s[5]:.2f}")
print(f"{'UKF':<10} | {ukf_s[2]:.2f} ± {ukf_s[3]:.2f} m {'':<14} | {ukf_s[4]:.
 ↪2f} ± {ukf_s[5]:.2f}")
print(f"{'PF':<10} | {pf_s[2]:.2f} ± {pf_s[3]:.2f} m {'':<14} | {pf_s[4]:.2f} ±␣
 ↪{pf_s[5]:.2f}")
print(f"{'APF':<10} | {apf_s[2]:.2f} ± {apf_s[3]:.2f} m {'':<14} | {apf_s[4]:.
 ↪2f} ± {apf_s[5]:.2f}")
print("="*120)

# Grafické porovnání (Boxplot Position Error)
plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 1)
plt.boxplot([agg_pos["BKN"], agg_pos["UKF"], agg_pos["PF"]], labels=['BKN',␣
 ↪'UKF', 'PF'], patch_artist=True, boxprops=dict(facecolor='lightblue'))
plt.title("Position Error [m]")
plt.grid(True, axis='y', linestyle='--', alpha=0.7)

# Grafické porovnání (Boxplot ANEES)
plt.subplot(1, 2, 2)
# Filtrujeme NaN pro boxplot
anees_data = [
    [x for x in agg_anees["BKN"] if not np.isnan(x)],
    [x for x in agg_anees["UKF"] if not np.isnan(x)],
    [x for x in agg_anees["PF"] if not np.isnan(x)]
]
plt.boxplot(anees_data, labels=['BKN', 'UKF', 'PF'], patch_artist=True,␣
 ↪boxprops=dict(facecolor='lightgreen'))
plt.axhline(y=4.0, color='r', linestyle='--', label='Ideal (4.0)')
plt.title("ANEES (Consistency)")
plt.legend()
plt.grid(True, axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```

```
=== VYHODNOCENÍ BKN NA TESTOVACÍ SADĚ (s ANEES) ===
Načítám data z: ./generated_data_synthetic_controlled/test_set/test.pt
Počet testovacích trajektorií: 20
Ensemble size (BKN): 100
Délka sekvence: 1000
Modely: BKN vs. UKF vs. PF vs. APF
```

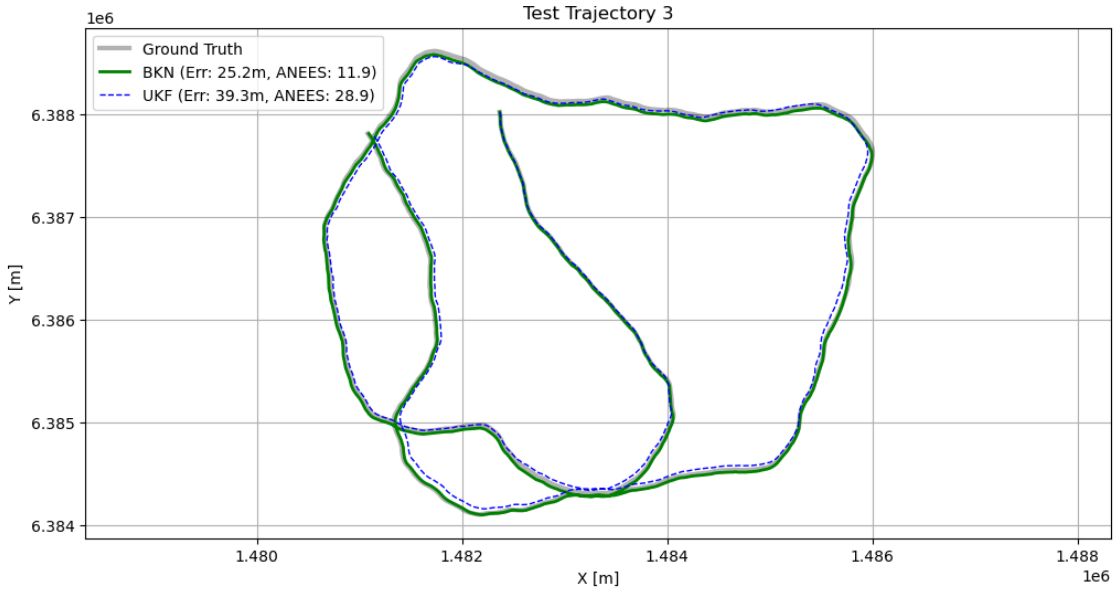
Evaluace: 0%| | 0/20 [00:00<?, ?it/s]



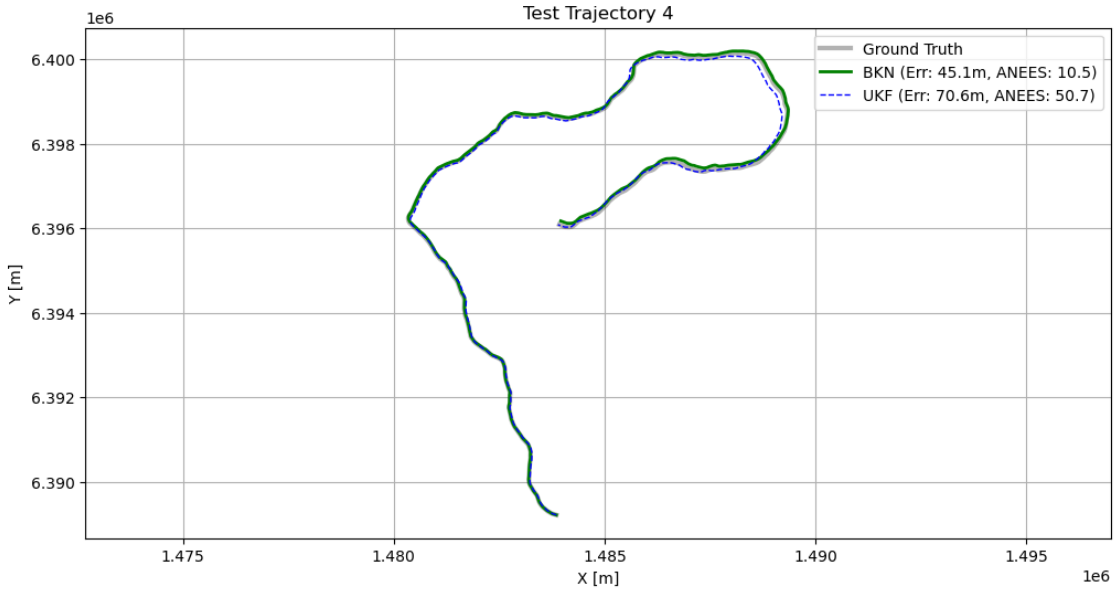
Evaluace: 5%| | 1/20 [00:06<01:57, 6.19s/it]



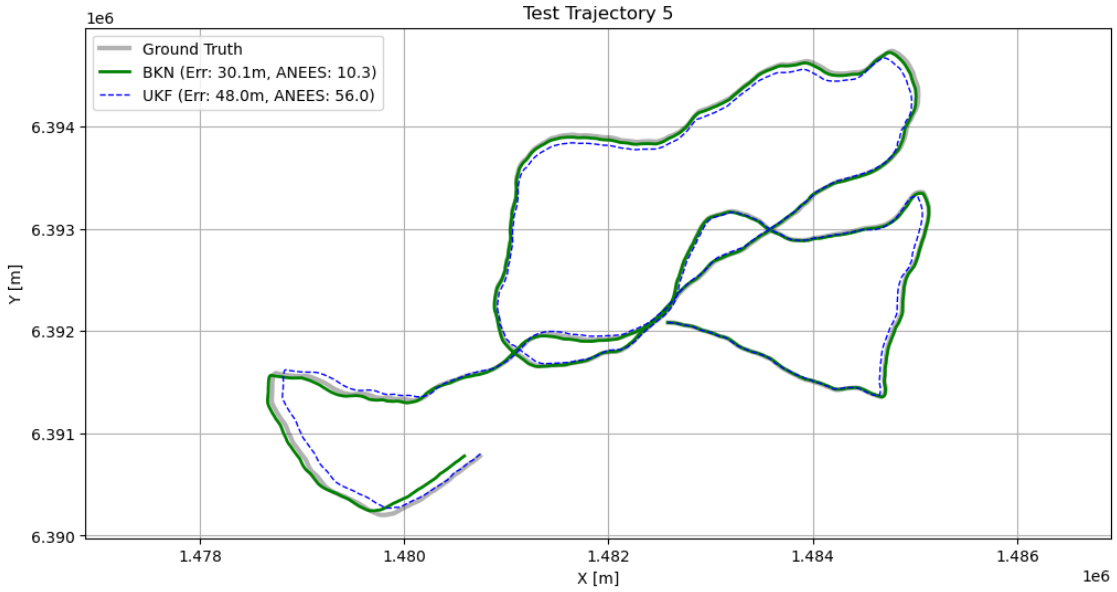
Evaluace: 10%| | 2/20 [00:11<01:46, 5.90s/it]



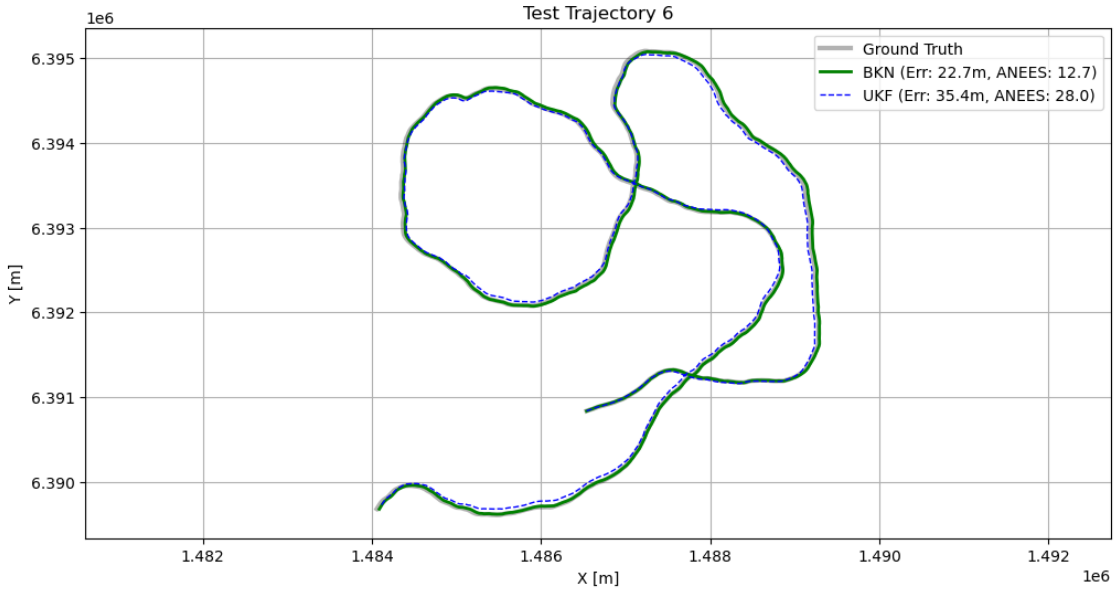
Evaluace: 15%| | 3/20 [00:18<01:42, 6.05s/it]



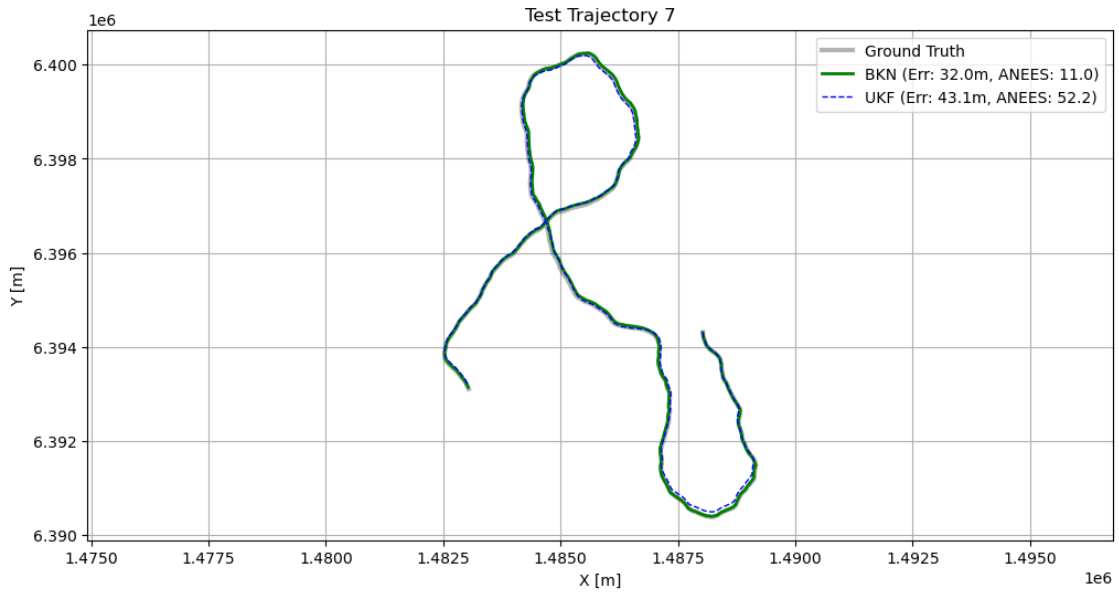
Evaluace: 20%| | 4/20 [00:24<01:36, 6.04s/it]



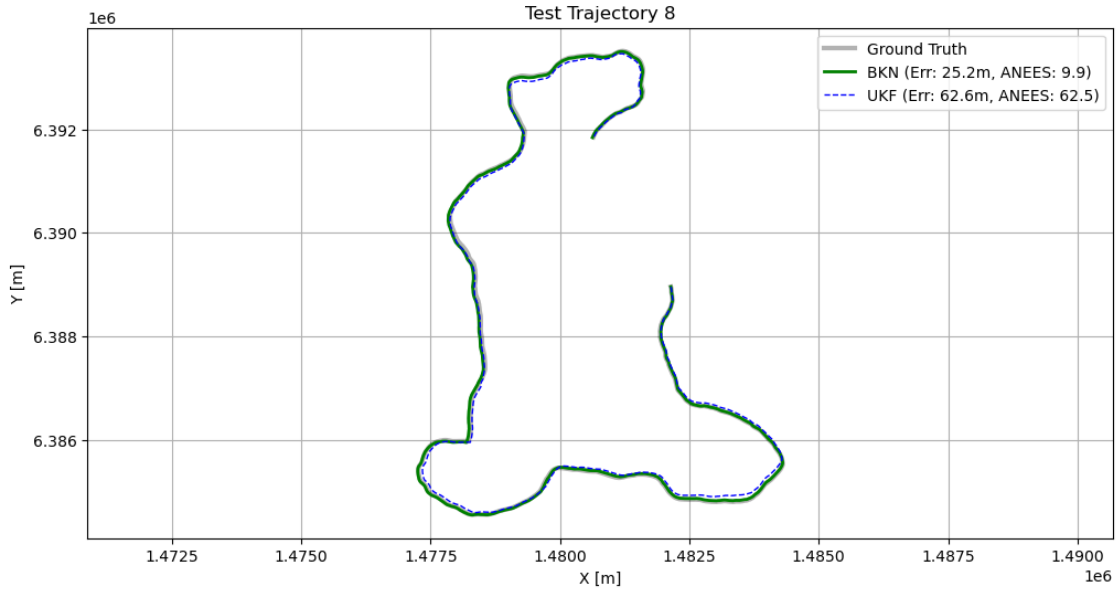
Evaluace: 25%| | 5/20 [00:30<01:30, 6.03s/it]



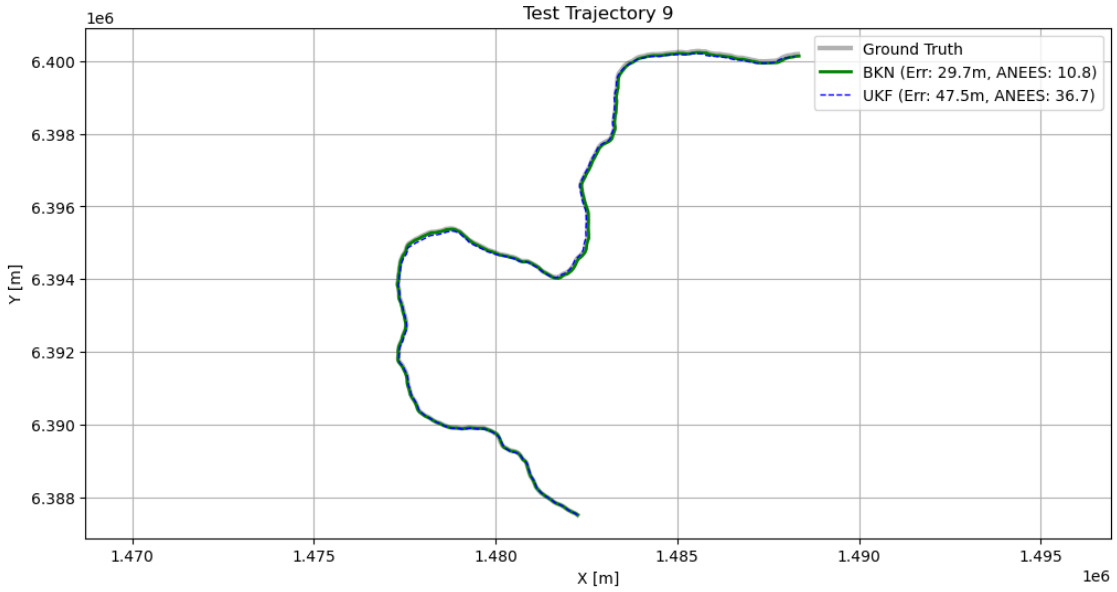
Evaluace: 30%| | 6/20 [00:35<01:23, 5.94s/it]



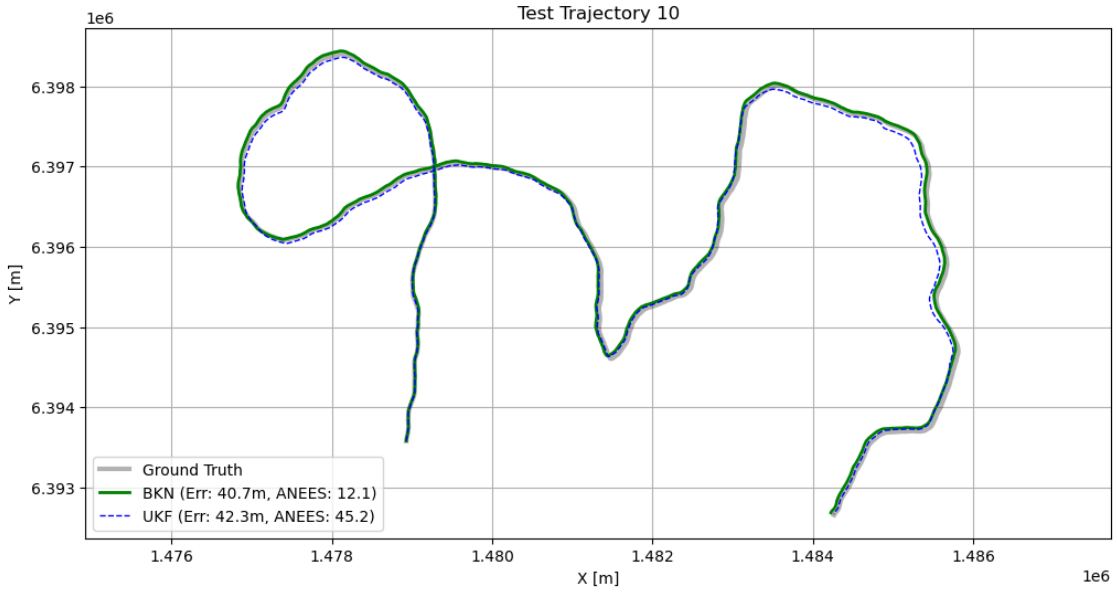
Evaluace: 35%| | 7/20 [00:41<01:17, 5.93s/it]



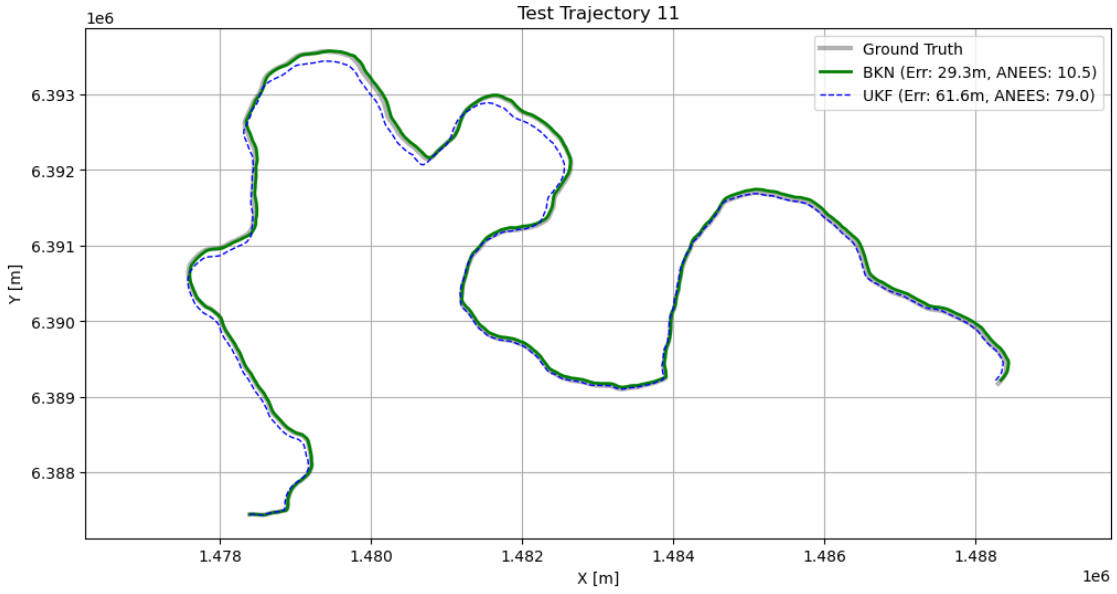
Evaluace: 40%| | 8/20 [00:47<01:11, 5.93s/it]



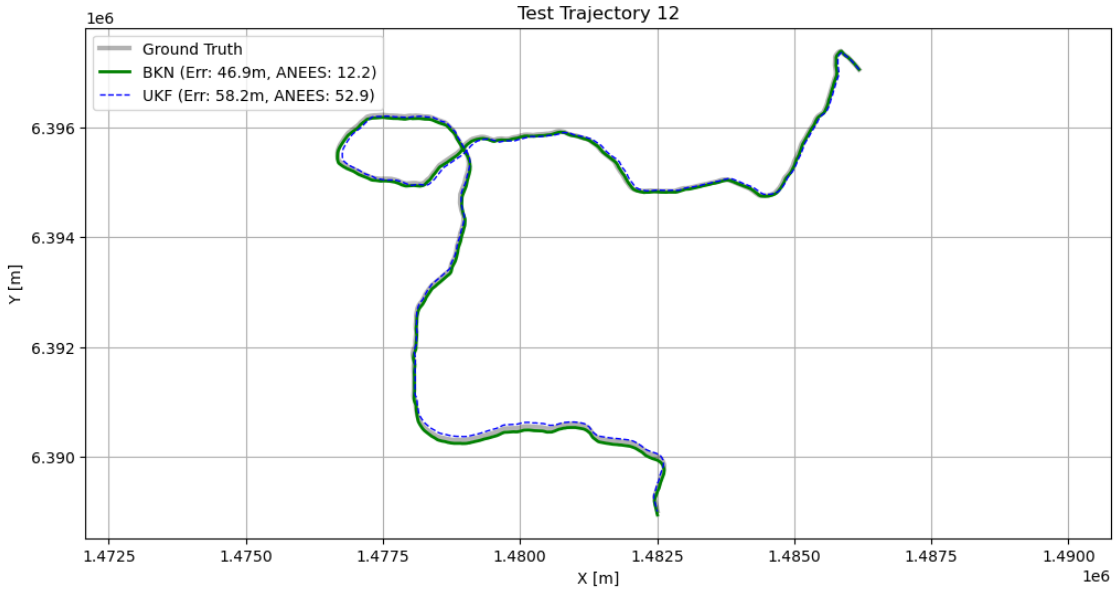
Evaluace: 45%| | 9/20 [00:53<01:05, 5.95s/it]



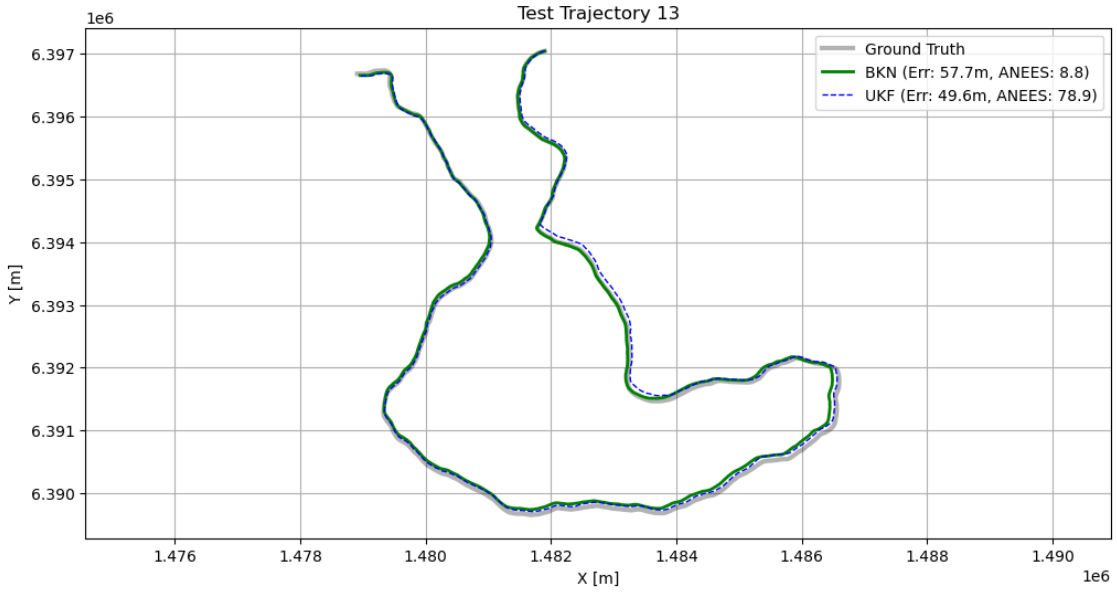
Evaluace: 50%| | 10/20 [00:59<00:59, 5.94s/it]



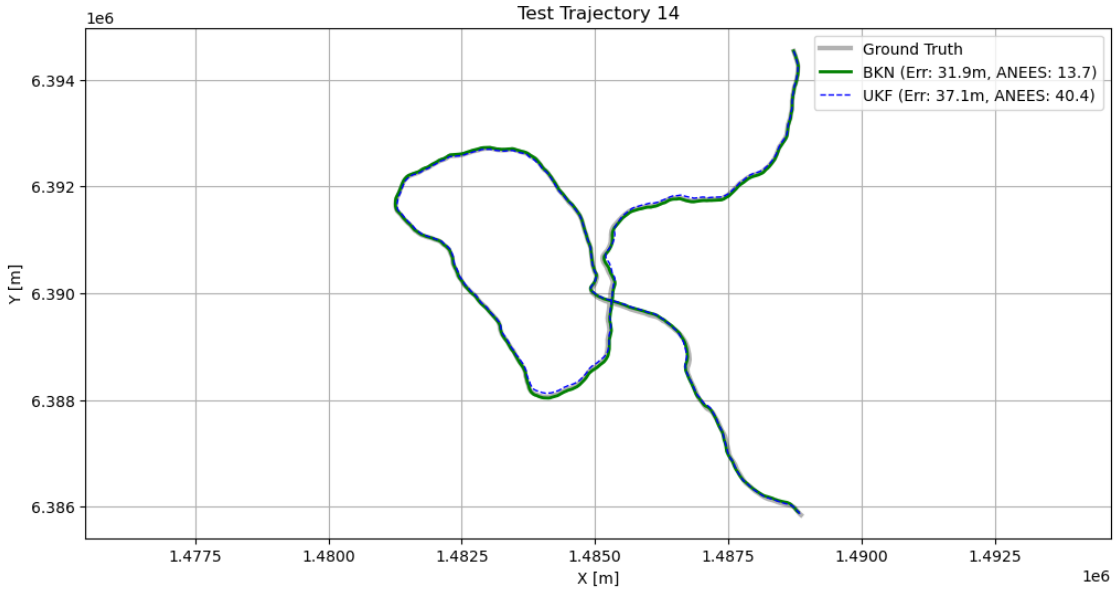
Evaluace: 55%| | 11/20 [01:05<00:53, 5.91s/it]



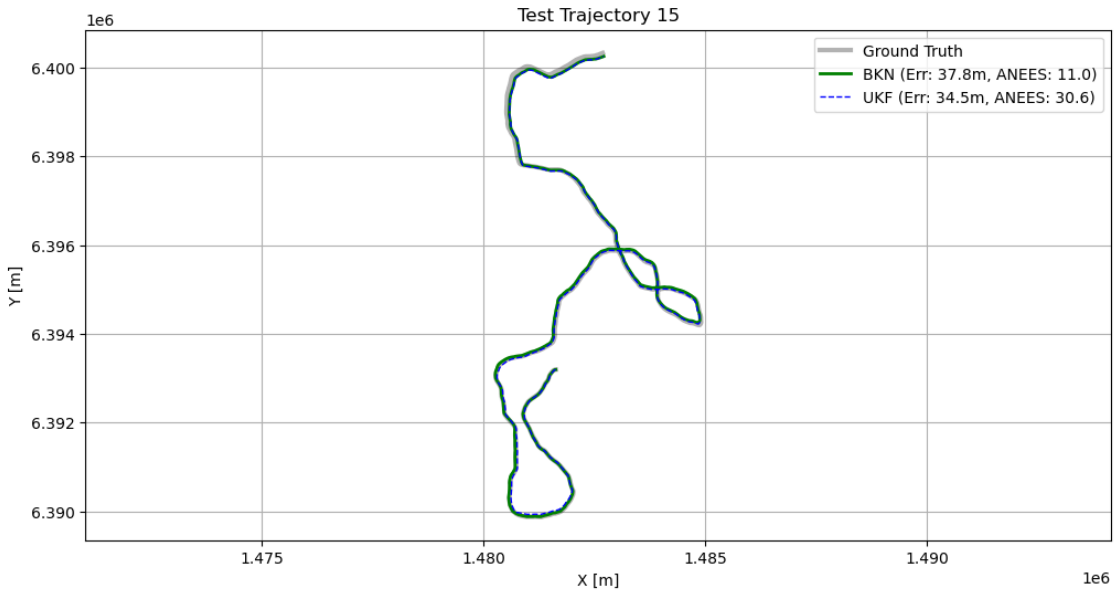
Evaluace: 60%| | 12/20 [01:11<00:47, 5.89s/it]



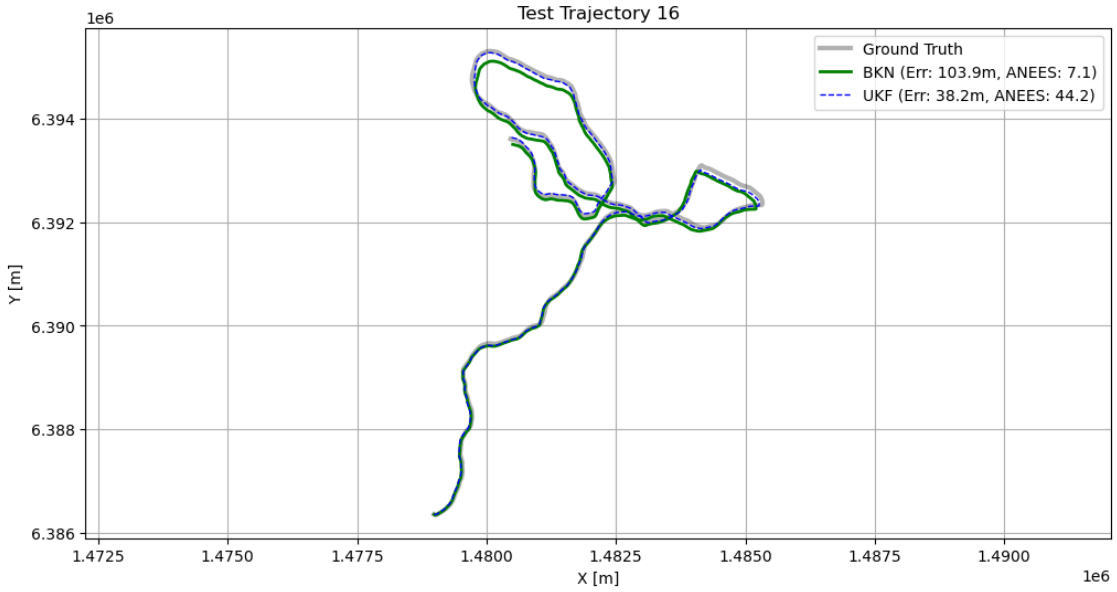
Evaluace: 65%| | 13/20 [01:17<00:40, 5.83s/it]



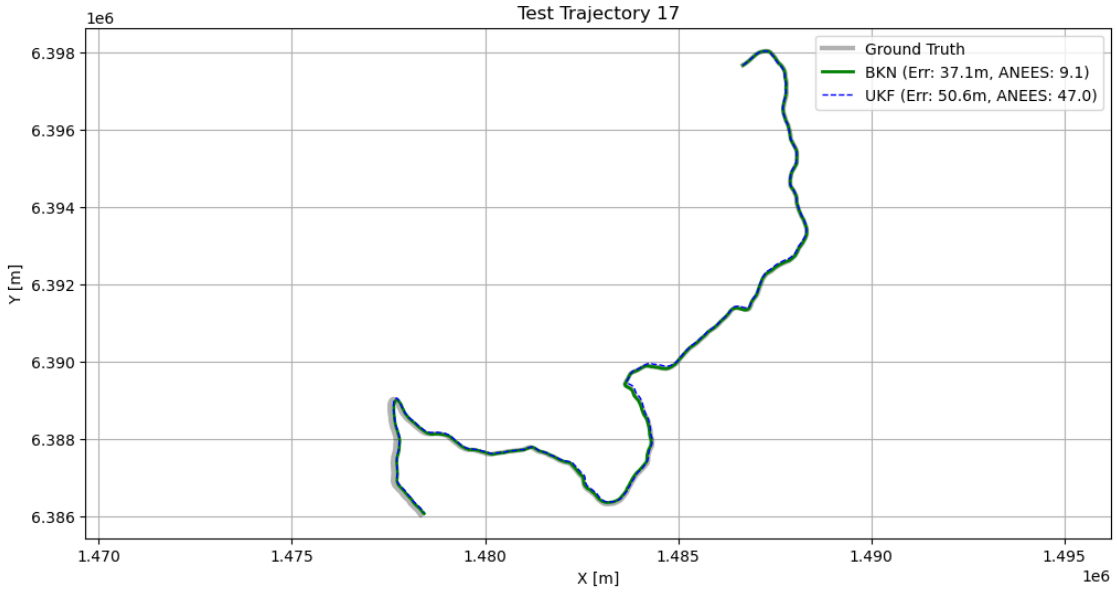
Evaluace: 70%| | 14/20 [01:23<00:35, 5.92s/it]



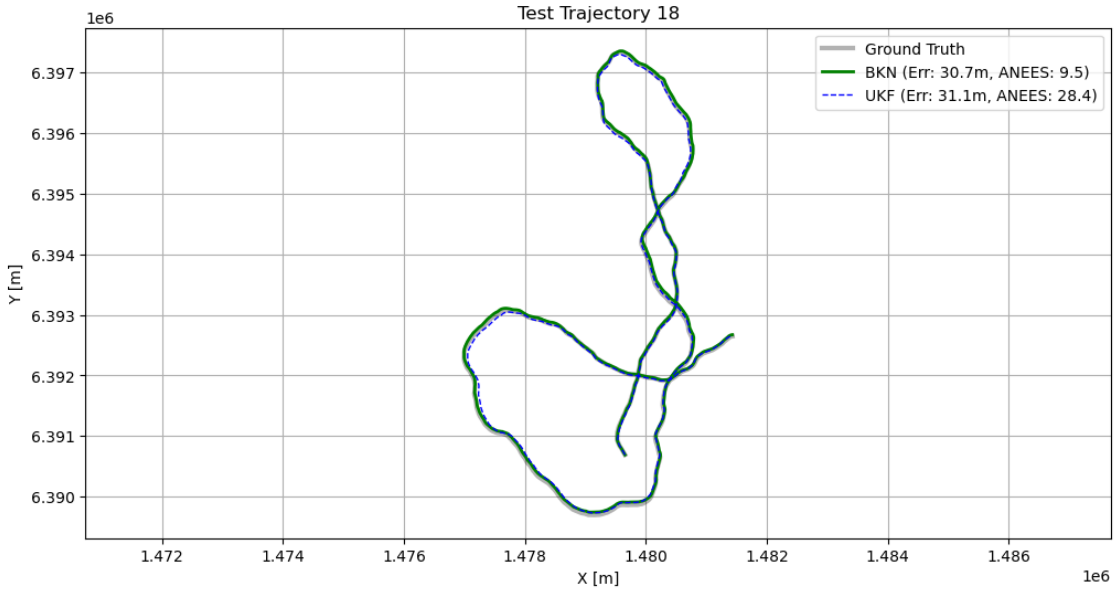
Evaluace: 75%| | 15/20 [01:29<00:29, 5.98s/it]



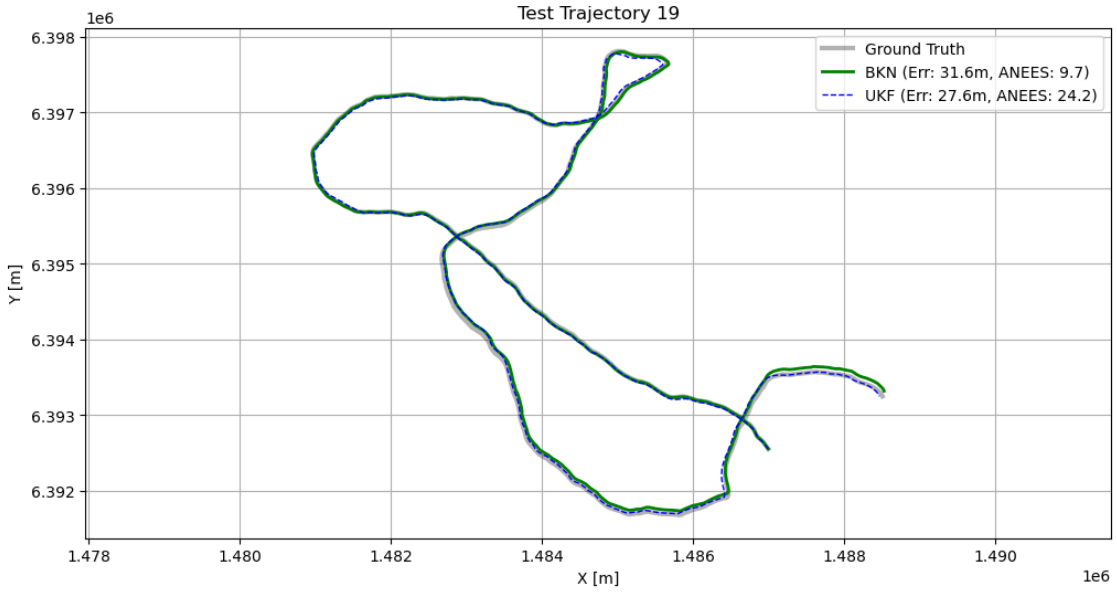
Evaluace: 80%| | 16/20 [01:35<00:23, 5.98s/it]



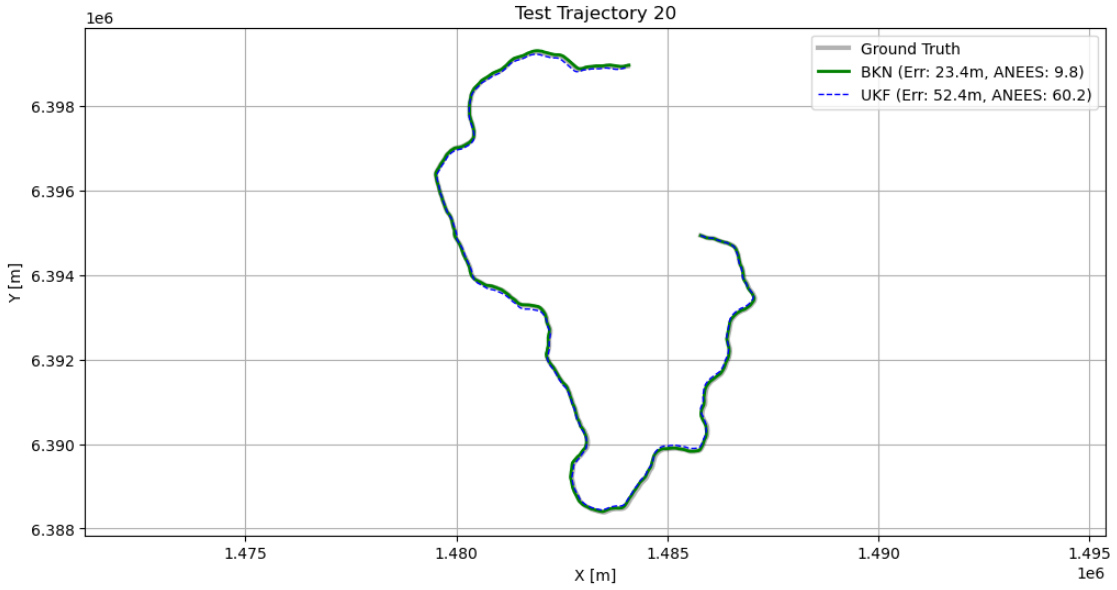
Evaluace: 85%| | 17/20 [01:42<00:18, 6.27s/it]



Evaluace: 90%| | 18/20 [01:48<00:12, 6.17s/it]



Evaluace: 95%| | 19/20 [01:54<00:06, 6.11s/it]



Evaluace: 100%| | 20/20 [01:59<00:00, 6.00s/it]

```
================================================================================
========================================
DETAILNÍ VÝSLEDKY (Pozice v metrech | ANEES - ideál ~4.0)
================================================================================
```

```
========================================
    Run_ID  BKN_PosErr  BKN_ANEES  UKF_PosErr  UKF_ANEES  PF_PosErr  \
0        1       40.21       9.59       36.47      27.49   3,907.06
1        2       25.44      10.79       42.93      50.69     687.46
2        3       25.16      11.89       39.30      28.94     268.53
3        4       45.14      10.48       70.63      50.69   7,997.62
4        5       30.07      10.29       48.03      56.02   7,362.50
5        6       22.70      12.67       35.42      28.00   3,789.49
6        7       31.96      11.01       43.06      52.23     277.89
7        8       25.20       9.88       62.63      62.46   1,945.47
8        9       29.69      10.82       47.46      36.72     102.16
9       10       40.70      12.12       42.29      45.23   3,001.36
10      11       29.30      10.50       61.62      78.99     659.19
11      12       46.92      12.16       58.15      52.91  13,889.70
12      13       57.67       8.79       49.65      78.87   7,384.99
13      14       31.88      13.66       37.09      40.42     409.53
14      15       37.82      11.02       34.46      30.61   9,296.38
15      16      103.88       7.09       38.24      44.20   6,520.79
16      17       37.12       9.05       50.59      47.01     642.30
17      18       30.65       9.49       31.14      28.38   7,309.58
18      19       31.62       9.67       27.60      24.15   3,766.81
19      20       23.37       9.82       52.39      60.23   2,734.85

    APF_PosErr
0        46.00
1        52.02
2        41.39
3        47.47
4        94.71
5        28.40
6        61.45
7       108.04
8        48.22
9        69.31
10      143.83
11      154.08
12      119.36
13       61.40
14       75.02
15      104.43
16      430.00
17       27.33
18    4,222.10
19       38.19

================================================================================
========================================
SOUHRNNÁ STATISTIKA (20 trajektorií)
```

```
================================================================================
========================================
Model      | Pos Error [m] (Mean ± Std)     | ANEES (Mean ± Std)
--------------------------------------------------------------------------------
BKN        | 37.32 ± 17.57 m                | 10.54 ± 1.46
UKF        | 45.46 ± 11.10 m                | 46.21 ± 15.77
PF         | 4097.68 ± 3727.72 m                | 716135155256074.12 ±
5527239534561218.00
APF        | 298.64 ± 904.15 m                | 116154200.00 ± 501530272.00
================================================================================
========================================
```