# JP Morgan MLCOE TSRL 2026 Internship Question 2:

# Particle Flow Filter and Differentiable Particle Filter

Modern banking and financial systems generate large volumes of sequential data, such as asset prices, interest rates, transaction records, and risk indicators. Modeling the underlying hidden state (latent dynamics) of such systems is critical for tasks like risk management, portfolio optimization, anomaly detection, and regulatory compliance.

**State Space Models (SSMs)** provide a powerful probabilistic framework to describe the evolution of latent states over time and their relationship with noisy observation. The classical **Kalman Filter (KF)** offers an optimal solution for linear-Gaussian SSMs. However, real-world financial systems often exhibit complex, nonlinear, and non-Gaussian behaviors, where Kalman filter is no longer adequate.

To tackle these challenges, practitioners have developed nonlinear non-Gaussian filters, among which the **Particle Filter (PF)** is a widely used sequential Monte Carlo method. While PF is flexible and broadly applicable, it suffers from issues such as particle degeneracy, sample impoverishment, and inefficiency in high-dimensional scenarios. **Particle Flow Filters (PFF)** have recently emerged as a promising alternative, which have been proposed to address these challenges.

This assignment will guide you from basic SSM filtering to advanced particle flow methods, culminating in an open-ended exploration of **Differentiable Particle Filters (DPF)** and HMC-based parameter inference. Your tasks are designed to both solidify your understanding of sequential inference and numerical stability of particle flows, and to enable end-to-end differentiable PF pipelines.

In your final report, please do make sure that you include a literature review of the methods that go beyond the methods discussed here to reflect your understanding of the problem.

**Part 1: From classical filters to particle flows**

Build up from basic filtering for SSMs to motivate the need for particle flow methods

1. Warm-up. For those who are not familiar with filtering, do spend some time learning these basics before the next part.
2. Linear-Gaussian SSM with Kalman Filter
3. Implement the Kalman filter for a multidimensional linear-Gaussian SSM.(Do not use tfp.distributions.LinearGaussianStateSpaceModel). Use synthetic data from a standard LGSSM, see examples 2 in [Doucet(09)].
4. Analyze its filtering optimality and numerical stability: compare filtered means/covariances to the Kalman recursion; use Joseph stabilized covariance updates; discuss conditioning number.
5. Nonlinear/Non-Gaussian SSM with EKF/UKF and Particle Filter
6. Design a nonlinear and non-Gaussian SSM, you can use a stochastic volatility model (example 4 in [Doucet(09)]) or nonlinear tracking models(e.g. Range-Bearing observation model)
7. Implement the Extend Kalman filter (EKF) and Unsent Kalman Filter(UKF), discuss linearization accuracy limits and sigma point failures under strong nonlinearity.
8. Implement a standard particle filter for your model. (Do not use the tfp.experimental.mcmc.particle\_filter). Visualize and discuss issues such as particle degeneracy.
9. Compare PF and EKF/UKF performance. How to evaluate your SSMs? What metrics can you use? In practice, we care about the runtime and memory, could you also compare the runtime and peak memory(CPU/GPU RAM) for each SSM?
10. Deterministic and Kernel Flows
11. Study the Exact Daum-Huang (EDH) flow and Local Exact Daum-Huang (LEDH) flow, (see [Daum(10)] and [Daum(11)]), and the invertible particle flow particle filter (PF-PF) framework (see Li(17)). Replicates the main results in Li(17).
12. Implement the kernel-embedded particle flow filter in an RKHS following Hu(21). Then compare the scalar kernel and diagonal matrix-valued kernel. Use experiments to demonstrate the matrix-valued kernel can prevent collapse of observed-variable marginals in high-dimensional, plot similar figures as figure 2-3 in Hu(21).
13. Compare EDH,LEDH, and kernel PFF on the SSM you designed in last particle filter question. Analyze when each method excels or fails(nonlinearity, observation sparsity, dimension, conditioning). Include stability diagnostics(flow magnitude, Jacobian conditioning).

**Part 2: Stochastic Particle Flow and Differentiable PF**

Introduce and evaluate stochastic particle flows for stability and design a differentiable PF pipeline with OT resampling.

1. Stochastic particle flow:
   1. Other than the deterministic particle flow, we can construct stochastic particle flow as shown in Dai(21) and Dai(22). Please replicate the main results of Dai(22)
   2. Repeating part 1, if we use the optimized particle flow of Dai(22) as the proposal for the particle flow particle filter of Li(17), does it improve the result of the LEDH particle flow particle filter of Li(17)
2. Differentiable particle filter with OT resampling
3. Implement a differentiable PF following Corenflos(21) and Chen(23).
4. Start with soft-resampling (mixture with uniform) to enable backpropagation through weights
5. Upgrade to entropy-regularized optimal transport (OT) resampling with Sinkhorn algorithm. Tune the regularization parameter and iterations for bias-variance-speed trade-off.
6. There are many other approaches for DPF (see Chen(23)), compare other algorithms(at least two) with OT resampling. Which metrics and diagnostics you can choose? Accuracy? Differentiability? Efficiency?

**Bonus question 1: HMC with invertible flows and differentiable resampling**

**Background:** Hamiltonian Monte Carlo is usually a much better method than Metropolis Hasting algorithm for MCMC as discussed in <https://arxiv.org/pdf/1206.1901> . You can find many useful functions in Tensorflow Probability and the experimental library. Unfortunately particle filters are not naturally differentiable, limiting their use in gradient-based Bayesian inference (e.g. HMC).

1. Consider the problem shown in section 3.1 of <https://www.stats.ox.ac.uk/~doucet/andrieu_doucet_holenstein_PMCMC.pdf> (equation 14 and 15). Please estimate the model using the invertible PF-PF of Li(17).
2. Incorporate entropy-regularized OT resampling in Corenflos(21), we can make the particle filter differentiable. Is it possible to apply the algorithm to Li(17) and apply HMC to the problem of the simple nonlinear state space model of section 3.1 shown in part a. of this question? Compare HMC with PMMH in terms of acceptance, chain ESS, runtime, and others.
3. Discuss advantages and challenges you encountered: differentiability-bias trade-off, OT regularization effects, gradient stability and variance, and others.

**Bonus Question 2: Neural Acceleration of OT resampling**

**Background:** Sinkhorn-based OT resampling can be costly. Solving repeatedly the optimal transport problem in bonus question 1 can be very time consuming. A natural idea is to use neural networks to approximate or accelerate optimization steps in OT.

1. Can we use the idea in Chaudhari(25) to avoid applying the Sinkhorn algorithm repeatedly? What additional input variables such as the model parameters and other necessary variables must we give to the neural network so that we can avoid learning the solution neural network repeatedly?
2. The algorithm, as shown in equation 4 of the paper, solves a partial differential equation. Does the theory of neural operator (see Jha(25)) apply? If it does, what method or architecture can we leverage to improve the training of the operator?

**Bonus Question 3: Particle-Flow Inference for Neural State-Space Models**

**Background:** State-Space LSTM(SSL) models combine LSTM transition with probabilistic emissions and use Particle Gibbs(PG) for joint posterior sampling, see Zheng(17).

1. Compare your DPF-HMC with particle Gibbs on example 1 and 2 in Zheng(17). What metrics can you use for comparison?
2. Summarize you final DPF method and write down the entire pipeline in detail. Which type of particle flow did you use? How exactly did you perform the resampling? What modifications did you make yourself? What techniques did you use? Do you think the current version is best? If you have more 3 months, how would you further optimize it to achieve a better DPF?

# References

[Daum(10)] Daum, Fred, Jim Huang, and Arjang Noushin. "Exact particle flow for nonlinear filters." *Signal processing, sensor fusion, and target recognition XIX*. Vol. 7697. SPIE, 2010.

[ Daum(11)] Daum, Fred, and Jim Huang. "Particle degeneracy: root cause and solution." *Signal Processing, Sensor Fusion, and Target Recognition XX*. Vol. 8050. SPIE, 2011.

[Li(17)] Li, Yunpeng, and Mark Coates. "Particle filtering with invertible particle flow." *IEEE Transactions on Signal Processing* 65.15 (2017): 4102-4116.

[Hu(21)] Hu, Chih‐Chi, and Peter Jan Van Leeuwen. "A particle flow filter for high‐dimensional system applications." *Quarterly Journal of the Royal Meteorological Society* 147.737 (2021): 2352-2374.

[Dai(21)] Dai, Liyi, and Fred Daum. "A new parameterized family of stochastic particle flow filters." *arXiv preprint arXiv:2103.09676* (2021).

[Dai(22)] Dai, Liyi, and Fred Daum. "Stiffness mitigation in stochastic particle flow filters." *IEEE Transactions on Aerospace and Electronic Systems* 58.4 (2022): 3563-3577.

[Corenflos(21)] Corenflos, Adrien, et al. "Differentiable particle filtering via entropy-regularized optimal transport." *International Conference on Machine Learning*. PMLR, 2021.

[Chaudhari(25)] Chaudhari, Shreyas, Srinivasa Pranav, and José MF Moura. "GradNetOT: Learning Optimal Transport Maps with GradNets." *arXiv preprint arXiv:2507.13191* (2025).

[Jha(25)] Jha, Prashant K. "From Theory to Application: A Practical Introduction to Neural Operators in Scientific Computing." *arXiv preprint arXiv:2503.05598* (2025).

[Doucet(09)] Doucet Arnaud, Johansen Adam. "A tutorial on particle filtering and smoothing: Fifteen years later." (2009).

[Chen(23)] Chen, Xiongjie, and Yunpeng Li. "An overview of differentiable particle filters for data-adaptive sequential Bayesian inference." *arXiv preprint arXiv:2302.09639* (2023).

[Zheng(17)] Zheng, Xun, et al. "State space LSTM models with particle MCMC inference." *arXiv preprint arXiv:1711.11179* (2017).