Optimization of Sampling Techniques for Particle Physics Datasets Using Fast Point Cloud Diffusion

[Hasif Ahmed][*]

Abstract

Our research investigates the optimization of sampling techniques applied to particle physics datasets, focusing on numerical methods and comprehensive data analysis. The unique challenges encompassed by particle physics data, such as continuous coordinates, stochastic dimensionality, and permutation invariance, distinguish it from standard datasets. The prevalent deep generative models, mainly designed for image data, fall short for these datasets. Our approach centers on a novel neural network simulation called Fast Point Cloud Diffusion (FPCD). The core aim is to boost the efficiency and speed of diffusion models, assessed via minimized Wasserstein Distances between authentic and generated data distributions and reduced sampling time.

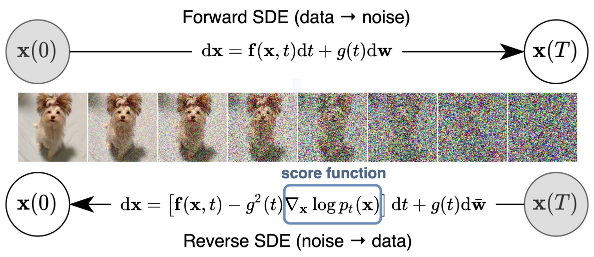
1 Background

Particle physics datasets, with their continuous coordinates and symmetries like permutation invariance, present challenges that traditional deep generative models often cannot meet. FPCD[1] estimates the gradients of the log-likelihood concerning data points, commonly referred to as the score function. This score function is pivotal, guiding the diffusion process to generate novel samples that mimic the underlying data distribution. To evaluate the efficiency of these sampling techniques, the Wasserstein Distance (W1) metric proves instrumental. The inherent denoising process in FPCD is intrinsically

[*]Lawrence University

stochastic, leading to the formulation of Stochastic Differential Equations (SDEs) that necessitate solving.

Wasserstein Metrics: The Wasserstein-1 (W1) metrics are a set of mathematical measures used to quantify the similarity or dissimilarity between probability distributions. In the context provided here, the W1 metrics are applied to assess the accuracy of different sampling methods for particle simulation across multiple particle types.



Forward SDE (data → noise)
$\mathbf{x}(0)$
$\mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\mathrm{d}t + g(t)\mathrm{d}\mathbf{w}$
$\mathbf{x}(T)$
score function
$\mathbf{x}(0)$
$\mathrm{d}\mathbf{x} = \left[\mathbf{f}(\mathbf{x}, t) - g^2(t)\boxed{\nabla_{\mathbf{x}} \log p_t(\mathbf{x})}\right]\mathrm{d}t + g(t)\mathrm{d}\bar{\mathbf{w}}$
$\mathbf{x}(T)$
Reverse SDE (noise → data)

Figure 1: Denoising yields a SDE[2]

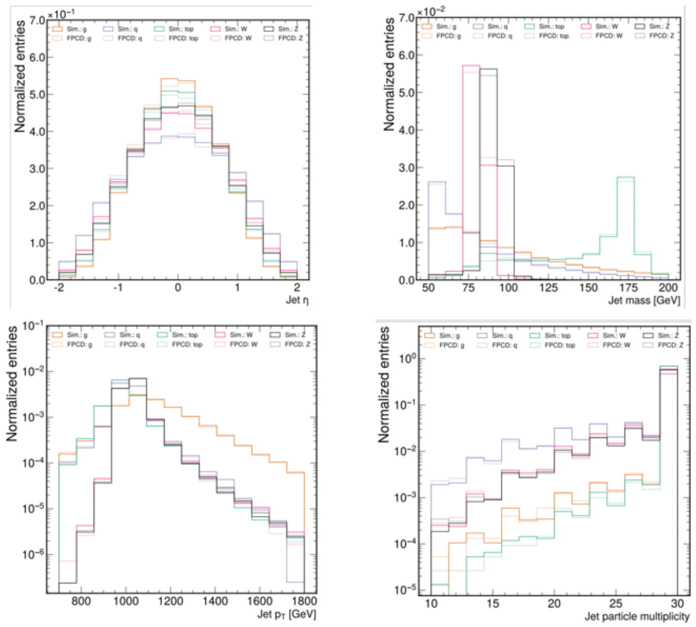


Figure 2: Particle metrics generated from FPCD sampling

In this context, the distributions represent the simulated particle properties for each particle type. The W1 metrics, including W1M, W1P, and W1EFP, capture different aspects of the distribution comparison. W1M evaluates the similarity between the means of the distributions, W1P measures the distance between percentiles, and W1EFP assesses the difference between the cumulative distribution functions.

2 Numerical Methods

2.1 ODE Solver

Upon transformation, the SDE provides a deterministic ordinary differential equation (ODE). We leverage the scipy RK45 and RK23 (Runge-Kutta) models to solve this ODE, presented as[1]:

$$dx_t = [f(x_t, t) - \frac{1}{2}g^2(t)\nabla_z \log \hat{p_\theta}(z_t)]dt$$

where $x \in$ some data distribution.

2.2 SDE Solver

The reverse formulation of the SDE is tackled using the Euler-Maruyama solver (EMSampler).

Recall the given reverse-time SDE:

$$d\mathbf{x} = [\mathbf{f}(\mathbf{x}, t) - g(t)^2\nabla_{\mathbf{x}} \log p_t(\mathbf{x})]dt + g(t)d\bar{\mathbf{w}}$$

Using the Euler-Maruyama scheme, the discretization with timestep Δt yields:

$$\mathbf{x}_{n+1} = \mathbf{x}_n + [\mathbf{f}(\mathbf{x}_n, t_n) - g(t_n)^2\nabla_{\mathbf{x}} \log p_{t_n}(\mathbf{x}_n)]\Delta t + g(t_n)\Delta\bar{\mathbf{w}}_n$$

Where:

- $\mathbf{x}_n$ is the approximation to the solution at time t_n.
- $\Delta\bar{\mathbf{w}}_n$ is the increment of the reverse Brownian motion over the interval Δt, and is approximated by a normal distribution with mean 0 and variance Δt.

3 Results and Analysis

Initially, we embarked on a comparison between diverse ODE libraries for the Initial value problem solver available in scipy.integrate library. We first look at the W1 metrics comparisons for RK23 and RK45 methods.

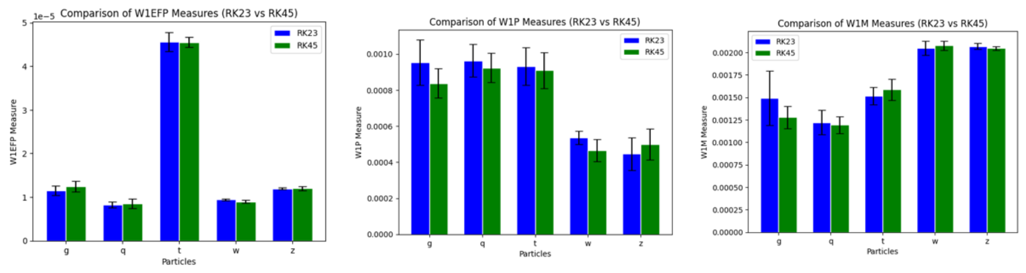


Figure 3: W1 metrics comparisons with error margins for RK23 and Rk45

Notice that the W1 metrics comparison is trivial.

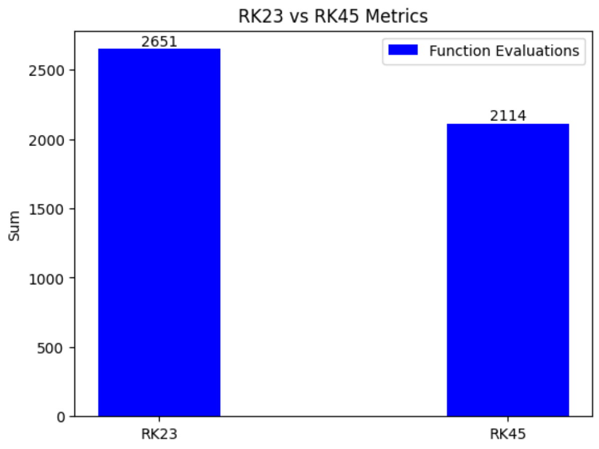


Figure 4: Function Evaluations comparisons for RK23 and Rk45

Also notice that without any consequential hikes in the W1 measures RK45 samples with requiring 537 less function evaluations than RK23 method (Figure 3), which is a 20 % acceleration.

We then want to look at different permutations of absolute and relative tolerance (atol and rtol) ranging from 10^{-5} to 10^{-3}. Once again- the goal is to find the method which gives faster sampling without any consequential hikes in W1 measures. We generate samples for atol=rtol cases and also for atol $\neq$ rtol cases. We compare their W1 performances and sampling time (reflected in the figures below):

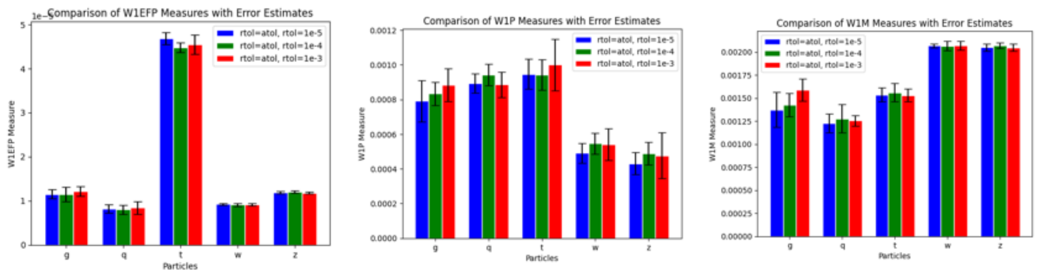


Figure 5: Rtol=Atol cases' W1 performances

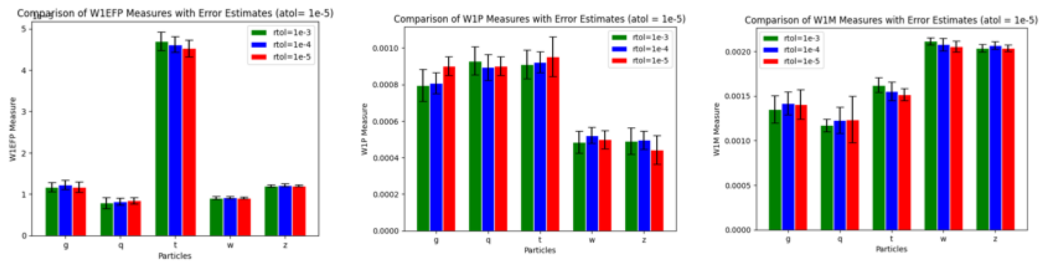


Figure 6: Rtol $\neq$ Atol cases' W1 performances

So we notice that RK45 (Rtol=Atol=10^{-3} has the fastest sampling without any huge bump in the W1 measures. With around required 600 function evaluations, this method is 4 times faster than our baseline RK45(Rtol=Atol=10^{-5}).

Now we want to consider solving the SDE directly (2.2) and compare its performance against the RK45 performances. Because we are writing the

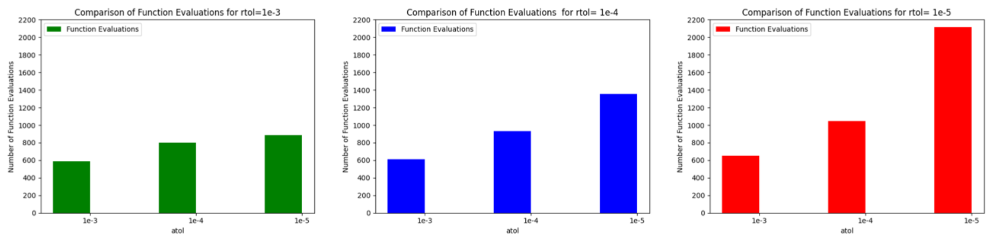


Figure 7: Number of Function evaluations for different tolerances in RK45

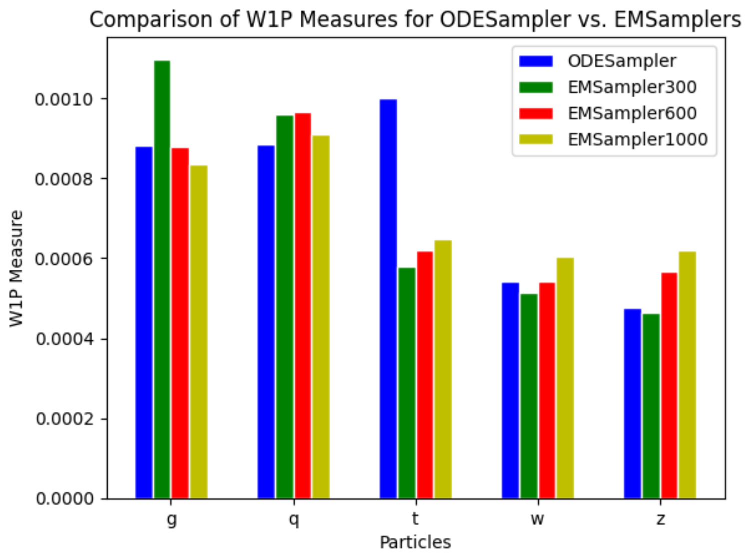


Figure 8: W1P performances for ODESamplers vs EMSamplers

solver on our own, now we can play around with the number of steps required to sample. That brings us to further comparisons:

Notice that the EMSampler's performance in comparable number of timesteps (300,600) perform almost comparable or worse compared to our ODESampler which is expected as the EMSampler without any corrector step has equivalent performance to blackbox RK45 solvers. So we want to go further and implement correctors.
Our exploration took another turn as we incorporated the corrector and predictor methodology. The corrector refines the predicted solution by leveraging local gradient details, ensuring the alignment of sampled paths with the intrinsic data distribution. We introduce the Signal-to-Noise Ratio (SNR) as a metric, quantifying the ratio of signal strength to the inherent noise. This acts as a compass, guiding the optimal dimension of the corrector step in our model.

We are simulating a stochastic differential equation (SDE) using TensorFlow. The SDE is of the general form:

$$dx = f(x)dt + g(x)dW$$

where dx is the change in x, $f(x)$ is the drift coefficient, $g(x)$ is the diffusion coefficient, dt is the time step, and dW is a brownian process which in our case is the noising/denoising process. [**2.2**]

First, a prediction for x is made, then a correction is applied using a Langevin MCMC (Markov Chain Monte Carlo) step.

1. **Predictor Step**:

   The predictor step is given by:

   $$x_{\text{updated}} = x - (f \cdot x - g^2 \cdot \text{score}) \cdot \text{Step size}$$

   and then

   $$x = x_{\text{updated}} + g \cdot \sqrt{|\text{step size}|} \cdot \text{Normal}(0, 1)$$

   where:

   - x is the current state.
   - f is the drift coefficient.

- g is the diffusion coefficient.
- Score is the gradient of some loss function with respect to x.
- Step size is the time step size.
- Normal$(0, 1)$ is a random number drawn from a normal distribution with mean 0 and standard deviation 1.

2. **Corrector Step (Langevin MCMC)**:

   The corrector step uses a Langevin MCMC method to update the predicted x. The Langevin MCMC method is used to sample from the posterior distribution of a model. It is a Metropolis-Hastings algorithm that proposes new states using the Langevin equation, which is a stochastic differential equation.

   The update rule for x in the Langevin step is given by:

$$\text{Langevin Step Size} = 2\left(\frac{\text{snr}\cdot|\text{noise}|}{|\text{grad}|}\right)^2\cdot\alpha$$

$$x_{\text{updated}} = x + \text{Langevin Step Size}\cdot\text{grad}$$

$$x = x_{\text{updated}} + \sqrt{2\cdot\text{Langevin Step Size}}\cdot\text{Normal}(0,1) + \text{Langevin Step Size}\cdot\text{grad}$$

   where:

   - grad is the gradient of some function with respect to x, by definition is the same as score.
   - $|$grad$|$ is the norm of grad.
   - $|$noise$|$ is the square root of the product of the dimensions of x.
   - snr is the Signal-to-Noise-Ratio, which we vary later.

By varying the SNR, we see the efficacy of the corrector step in terms of more accurate sampling. For a 1000 time steps for our stochastic samplers, we see how adding a corrector step performs better in W1 performances across most particles (Pictured below), which means better accuracy without added computational expense. Win-win!

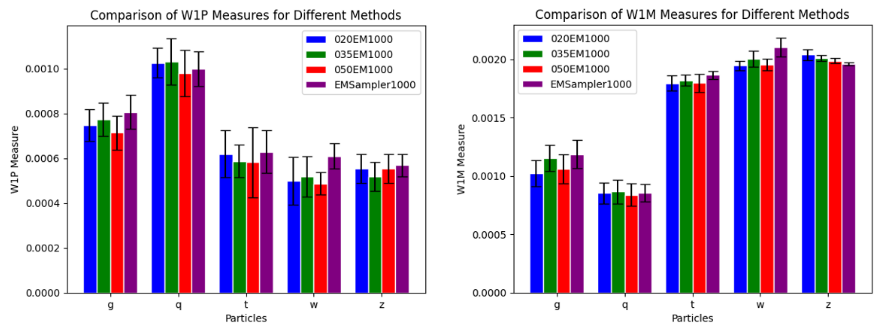


Figure 9: Corrected Sampler for various SNR values vs EMSampler1000 (without any corrector)

Now the goal becomes to lower the number of steps and permuting the SNR value till our accuracy performance is better or comparable to our ODE-Sampler's benchmark accuracy. Here we implement the idea of computing the average of W1 metrics across all 5 particles. The argument is as follows:

Let M be the set of all particle types, and let M_i be the ith particle type within this set. Each particle type M_i has an associated distribution P_i representing its simulated properties obtained from a particular sampling method. Similarly, let Q_i represent the expected or true distribution of particle type M_i.

The Wasserstein-1 distance $W_1(P_i, Q_i)$ between the distributions P_i and Q_i measures their dissimilarity. A lower W_1 distance implies a higher degree of similarity between the distributions.

We have three metrics for comparison: W_1^{M}, W_1^{P}, and W_1^{EFP}, which respectively represent the Wasserstein-1 distances calculated based on means, percentiles, and cumulative distribution functions.

To find the method that provides the most accurate particle simulation, we can compute the average of the three W_1 metrics for each particle type M_i:

$$\text{Average}_i = \frac{W_1^{\text{M}}(P_i, Q_i) + W_1^{\text{P}}(P_i, Q_i) + W_1^{\text{EFP}}(P_i, Q_i)}{3}$$

Now, for a specific sampling method, let's denote the average of W_1 metrics for all particle types as A_{method}:

$$A_{\text{method}} = \frac{\sum_i \text{Average}_i}{|M|}$$

Where $|M|$ represents the total number of particle types. Lower values of A_{method} indicate higher overall accuracy across particle types.

The method with the minimal A_{method} will have the smallest average W_1 metrics across all particle types, demonstrating the closest agreement between the simulated and expected distributions, and thus is the most accurate method according to this comparison approach. The python implementation for our computation can be found here.

Through this computation process and checking the results for various number of steps, we came to conclude that with SNR=0.165 and 250 timesteps, we find the stochastic solver performs better on both accuracy and speed compared to our RK45 ODESampler:

Particle Type	Method with Minimal Average W_1 Metrics
g	0165EM250
q	0165EM250
t	RK45 Benchmark
w	0165EM250
z	RK45 Benchmark

Table 1: Method with Minimal Average W_1 Metrics for Each Metric

Remarkably, this is 8.4 times swifter than our baseline ODESampler.

4 Conclusions

The Fast Point Cloud Diffusion (FPCD) model introduces a novel method for enhancing sampling techniques, particularly tailored to the intricate challenges posed by particle physics datasets. Through our investigations, it became evident that utilizing a stochastic solver in the denoising process offers considerable advantages over traditional blackbox Python solvers. Notably, these advantages encompass improvements in both accuracy and sampling speed.

Our proprietary stochastic solver, denoted as 0165EM250, outperforms the benchmark RK45 ODE solver by a factor of 8.4 in terms of speed, while maintaining a comparable level of accuracy. These findings highlight the substantial efficiency gains achievable with FPCD-based models.

Future research avenues will focus on analyzing the W1 variance across different particles, contingent on varying corrector sizes. Furthermore, we intend to delve into higher-order corrections and experiment with alternative Monte Carlo methods, aiming to ascertain their potential impact on enhancing sampling performance. Ultimately, refining the corrector steps stands out as a promising strategy to further accelerate the sampling process without sacrificing accuracy.

5 References

1. Mikuni et al. 2023, Fast Point Cloud Generation with Diffusion Models in High Energy Physics.

2. Song et al. 2020, Score-Based Generative Modeling through Stochastic Differential Equations.