`QMC_sampler` – A Quasi Monte Carlo Sampling Library

Documentation

April 4, 2025

1 Introduction

This document provides an overview of the `QMC_sampler` Python library, which generates Quasi Monte Carlo (QMC) point sets from a high-dimensional generating vector. These points are used in Multilevel Monte Carlo (MLMC) simulations, particularly for solving McKean-Vlasov type Stochastic Differential Equations (SDEs) like the Kuramoto oscillator model.

2 The Kuramoto Oscillator McKean-Vlasov SDE

The Kuramoto model describes a system of coupled oscillators, where each oscillator's phase evolves according to an SDE incorporating mean-field interactions. The McKean-Vlasov formulation extends the classical Kuramoto model by including stochastic perturbations. The governing equation is given by:

$$dX_t^i = \nu^i dt + \frac{K}{P} \sum_{j=1}^{P} \sin(X_t^j - X_t^i) dt + \sigma dW_t^i, \quad i = 1, \ldots, P, \tag{1}$$

where:

- X_t^i is the phase of the i-th oscillator at time t.
- ν^i is the frequency of the i-th oscillator.
- K is the coupling strength between oscillators.
- σ is the noise intensity.
- W_t^i is a Wiener process modeling stochastic fluctuations.

This SDE is simulated numerically using Euler-Maruyama and Richardson extrapolation methods described in later sections.

3 Class `QMC_sampler`

The `QMC_sampler` class manages QMC point sets generated from a predefined high-dimensional generating vector. This vector is typically loaded from a file and used to generate lattice rule points for simulations. It supports hierarchical sampling through an adaptive level structure and ensures efficient low-discrepancy point generation.

3.1 Attributes

- `points` (list of numpy arrays): Stores generated QMC points at different levels. Each entry is a matrix of shape $(2^{\text{level}}, d)$ where d varies per level.
- `max_level` (int): The current maximum refinement level.
- `generating_vector` (numpy array of shape $(d,)$): Stores the initial high-dimensional generating vector loaded from a file.
- `gen_vecs` (list of numpy arrays): Stores the history of constructed generating vectors for different levels.

3.2 Methods

3.2.1 `initialize_from_file(filename)`

- Reads a file containing a generating vector.
- The file should contain two columns: the dimension index and the corresponding vector component.
- Sorts the data by the dimension index and stores the generating vector.
- Resets the `max_level` attribute to zero.
- **Arguments:**
  - `filename` (str): Path to the input file.

3.2.2 `add_level()`

- Extracts a subvector from the generating vector and generates new QMC lattice rule points.
- Updates the list of points and increments the `max_level` counter.

4 Function Reference

4.1 `bridge_map(point, t, T)`

- Implements Brownian bridge construction.
- Uses midpoint recursion to generate Wiener process paths.
- **Arguments:**
  - `point` (numpy array of shape (M, P)): Input QMC points.
  - `t` (float): Initial time.
  - `T` (float): Final time.

4.2 `pts_to_paths(pts, shifts, T, incr=True)`

- Maps QMC points to Wiener paths and model parameters for Kuramoto oscillators.
- Computes a probability measure transform based on Gaussian quantiles.
- Applies the `bridge_map` function for path generation.
- **Arguments:**
  - `pts` (numpy array of shape (M, d)): Input QMC points.
  - `shifts` (numpy array of shape (m, d)): Shift vectors for each level.
  - `T` (float): Final time for simulation.
  - `incr` (bool, default=True): Whether to compute increments or absolute paths.
- **Returns:**
  - `p_measure_pts` (numpy array of shape (M, d)): Transformed QMC points representing Wiener paths and parameters.

4.3 `euler_maruyama_kuramoto(X0, nu, dW, T, P, sigma)`

- Implements the Euler-Maruyama discretization for the McKean-Vlasov Kuramoto system.
- Simulates the Kuramoto oscillator system with stochastic perturbations and mean-field interactions.
- **Arguments:**
  - `X0` (numpy array of shape (M, P)): Initial phase angles of the oscillators for M samples of P oscillators.

- `nu` (numpy array of shape (M, P)): frequencies for the M samples of P oscillators.
- `dW` (numpy array of shape (M, P, N)): Wiener increments for the M samples over N time steps.
- `T` (float): Final time of the simulation.
- `P` (int): Number of oscillators (particles).
- `sigma` (float): Noise intensity.

- **Returns:**
  - `X_T` (numpy array of shape (M, P)): Final phase angles at time T for each sample.

4.4 `richardson_euler_maruyama_kuramoto(X0, nu, dW, T, P, sigma)`

- Computes a Richardson extrapolated version of the Euler-Maruyama method for the McKean-Vlasov Kuramoto system.
- Uses fine and coarse discretizations to improve the accuracy of the simulation.
- **Arguments:**
  - `X0` (numpy array of shape (M, P)): Initial phase angles of the oscillators for M samples of P oscillators.
  - `nu` (numpy array of shape (M, P)): frequencies for the M samples of P oscillators.
  - `dW` (numpy array of shape (M, P, N)): Wiener increments for the M samples over N time steps.
  - `T` (float): Final time of the simulation.
  - `P` (int): Number of oscillators (particles).
  - `sigma` (float): Noise intensity.
- **Returns:**
  - `X_T` (float): Richardson-extrapolated final phase angle at time T.