Towards Better Estimation of Virtual Waiting Time in Non-Stationary Systems

Shaozhe Ke Zihao Ma
School of Management and Engineering, Nanjing University
221870264@smail.nju.edu.cn, 221870001@smail.nju.edu.cn

August 17, 2025

Abstract

This report tries to explore methods for achieving better and more cost-efficient estimation of virtual waiting time in non-stationary systems. Building on the k-nearest-neighbor (k-NN) based virtual waiting time estimation framework of Lin et al. [4], this report investigates two methodological extensions aimed at improving estimation efficiency and accuracy. First, we explore incorporating topological data analysis (TDA) [8] to capture latent structural patterns in simulation output. Second, we develop an adaptive k-NN strategy that dynamically adjusts the neighborhood size to balance bias and variance. In addition, we implement the Kolmogorov forward equations(KFEs) for queueing systems to compute the true waiting time distribution, providing a benchmark for performance evaluation. These enhancements are evaluated with the goal of identifying more computationally efficient or accurate approaches compared to the original method.

Contents

1 Introduction

Virtual waiting time is a fundamental performance measure in queueing systems, especially in non-stationary environments where arrival or service rates vary over time.

Accurately estimating virtual waiting time enables system designers and operators to assess service levels, allocate resources, and make informed operational decisions. However, non-stationarity introduces significant challenges for estimation, as classical steady-state or long-run average analysis often fails to capture transient behaviors.

Lin et al. [4] proposed a k-nearest-neighbor (k-NN) based framework for estimating virtual performance measures, including virtual waiting time, from retained simulation sample paths. Their method avoids rerunning simulations for each time point of interest, offering flexibility and efficiency compared to traditional on-the-fly computation. The approach also incorporates cross-validation to select an optimal neighborhood size k, balancing bias and variance in the estimator.

While the method in Lin et al. [4] provides a strong foundation, there remain opportunities to improve the accuracy or computational efficiency of virtual waiting time estimation especially in scenarios where the available sample size is limited. In this report, we explore two complementary directions for enhancement. First, we investigate the use of *topological data analysis* (TDA) to identify structural patterns in simulation outputs that may inform more effective neighborhood selection and enable faster classification. Second, we develop an *adaptive k-NN* scheme that dynamically adjusts k based on local data density and variability, potentially reducing bias in sparse regions and variance in dense regions.

Very recently, Topological Data Analysis (TDA) [8] has gained significant attention as a versatile tool for extracting and analyzing the intrinsic shape of data. Its applications span a wide range of domains, including robust optimization [5], where TDA-based features can enhance model stability under distributional shifts; data analysis [8], where topological summaries capture structural patterns beyond conventional statistical descriptors; and machine learning [2], where TDA facilitates the integration of heterogeneous data sources and multiparameters by revealing common topological structures. As outlined in Wasserman [8], TDA provides a rigorous framework for extracting, summarizing, and analyzing the topological features of data. This framework includes key concepts such as simplicial complexes, persistent homology, and stability theorems, which together enable the quantification of data shape in a robust and scalable manner. In this work, inspired by the general framework in Wasserman [8] but deviating from the k-nearest neighbor approach in Lin et al. [4], we construct a kernel function directly from the simulation output data of non-stationary queueing systems. This kernel is designed to capture intrinsic topological signatures—extracted via persistent homology—of the underlying system states. By embedding these features into a reproducing kernel Hilbert space (RKHS), we perform mode-seeking to identify representative states that are most informative for virtual waiting time estimation. This approach leverages the stability and robustness of TDA kernels while avoiding the local-sample-dependence of k-NN, aiming to achieve higher computational efficiency compared to the high complexity in Lin et al[4].

In addition to methodological extensions, we implement the *Kolmogorov forward equations* for selected queueing models to compute the exact distribution of true waiting times. This serves as a benchmark for evaluating the performance of the proposed estimators and for quantifying potential efficiency gains over the original k-NN framework.

The remainder of this report is organized as follows. Section **??** reviews the k-NN method of Lin et al. and related work on virtual waiting time estimation. Section **??**

presents the proposed TDA-based and adaptive k-NN enhancements, along with the Kolmogorov forward equations implementation. Section **??** reports numerical results and comparative evaluations. Finally, Section **??** summarizes the findings and discusses directions for future research.

2 k-NN in Virtual Waiting Time Estimation [4]

In this section, we aim to provide a comprehensive restatement of the methodology and results presented in Lin et al. [4], reproducing their key ideas, assumptions, and findings in our own words to ensure a clear understanding and to set the stage for our subsequent modifications and extensions. In Lin et al. [4], virtual waiting time estimation in non-stationary queueing systems is approached via a k-nearest neighbor (k-NN) methodology. The core idea is to estimate the virtual waiting time at a given system state by averaging the observed waiting times from its k most similar historical states, as measured in the space of chosen covariates (e.g., system time, arrival rate, service rate, queue length).

2.1 Historical Context and Motivation

In the early days of simulation language development, both memory—dynamic and persistent—and processing power were scarce resources. As a result, it was necessary to compute and compactly summarize performance statistics *on the fly*. This approach aligned well with the then-prevailing emphasis on predetermined long-run average performance. However, with modern computing capabilities, this perspective has become outdated and increasingly restrictive.

2.2 Defining Virtual Waiting Time

A common example of $V(\tau_0)$ is the virtual waiting time experienced by a customer arriving to a service system at time τ_0. However, this notion is less straightforward than it may initially appear, as multiple definitions exist. Specifically, the "virtual waiting time" at τ_0 can be interpreted as:

1. *Injected*: the waiting time of a hypothetical customer artificially inserted into the nominal stochastic arrival process.
2. *Phantom*: the amount of work ahead of a fictitious arrival at τ_0 that does not actually enter the system.
3. *Conditional*: the waiting time faced by an actual arrival from the nominal stochastic process, conditional on such an arrival occurring at time τ_0.

In certain cases, the injected and phantom definitions coincide; however, they may differ in systems with non-first-in-first-out (non-FIFO) disciplines or in networks where overtaking is possible, meaning customers may depart in a different order than they arrive. In this work, we focus on the *conditional* definition, which will be formally stated later.

2.3 The k-NN Estimation Framework

The k-NN approach treats the estimation of $V(\tau_0)$ as a regression problem in the covariate space. Let $\mathbf{X}_t$ denote the vector of covariates representing the system state at time t, and let Y_t be the observed waiting time for an arrival at t. For a query point $\mathbf{x}_0$, corresponding to the state at τ_0, the estimator is:

$$\hat{V}_k(\mathbf{x}_0) = \frac{1}{k} \sum_{i \in \mathcal{N}_k(\mathbf{x}_0)} Y_i,$$

where $\mathcal{N}_k(\mathbf{x}_0)$ denotes the set of k nearest neighbors of $\mathbf{x}_0$ in the historical dataset, with distance measured by a chosen metric (typically Euclidean).

Distance Metric and Feature Selection. The choice of covariates is crucial for performance. In Lin et al. [4], candidates include:

- Current simulation time t
- Instantaneous arrival rate $\lambda(t)$
- Service rate $\mu(t)$
- Current queue length $Q(t)$

The Euclidean distance in this feature space is then normalized to ensure comparability between dimensions.

2.4 Computational Procedure

The algorithm can be summarized as follows:

1. Collect a historical database of $(\mathbf{X}_t, Y_t)$ pairs during the simulation.
2. Normalize each feature to zero mean and unit variance.
3. For a query state $\mathbf{x}_0$, compute distances to all historical states.
4. Select the k nearest states and average their Y values to estimate $\hat{V}_k(\mathbf{x}_0)$.

2.4.1 Selection of Optimal k: Leave-One-Replication-Out Cross-Validation.

To select the optimal k in practice, Lin et al. [4] employ a Leave-One-Replication-Out Cross-Validation scheme, where one simulation replication is held out as a test set while the rest serve as the training set. The following pseudocode describes the process.

Algorithm 1 kNN Estimation via LORO-CV

Input: Search range $k_L < k_U$, NN = "nearest neighbors"
for $j = 1, 2, \ldots, n$ **do** ▷ Loop through all replications
    $S_{\text{test}} \leftarrow \{W(t_{ij}), t_{ij}; i = 1, 2, \ldots, M_j\}$ ▷ Set aside one replication as test set
    $S_{\text{train}} \leftarrow$ all data except S_{test} ▷ Use remaining data as training set
    Find k_U nearest neighbors in S_{train} for each $t_{ij} \in S_{\text{test}}$
    Store indices of k_U nearest neighbors into $\mathbf{M}_{\text{ind}} \in \mathbb{R}^{|S_{\text{test}}| \times k_U}$
end for
for $k = k_L$ to k_U **do** ▷ Search for optimal k
    Extract the first k columns from $\mathbf{M}_{\text{ind}}$
    Find k-nearest neighbors for each $t_{ij} \in S_{\text{test}}$ and compute $\hat{W}(t_{ij}, k)$
end for
for $k = k_L$ to k_U **do** ▷ Compute the EMSE for each k
    Compute the empirical mean squared error (EMSE):

$$\text{EMSE}(k) = \frac{\sum_{j=1}^{n} \sum_{i=1}^{M_j} \left[W_{ij} - \hat{W}(t_{ij}, k) \right]^2}{\sum_{j=1}^{n} M_j}$$

end for
Output: k^* that minimizes EMSE(k)

2.5 Experimental Setup in Lin et al. [4]

The authors validate their method in a variety of non-stationary queueing environments, such as $M_t/G/1$ and $M_t/M_t/1$ queues with time-varying arrival and service rates. Simulation parameters are selected to mimic realistic load fluctuations.

Table 1: Example parameter settings for simulation experiments

Experiment	Arrival Rate Function	Service Rate	Capacity
1	$\lambda(t) = 5 + 3\sin(0.5t)$	$\mu = 8$	∞
2	$\lambda(t) = 2 + 1.5\cos(0.3t)$	$\mu = 4$	10

2.6 Results and Observations

The main findings reported include:

- k-NN estimators outperform naive averaging in scenarios with significant time-varying behavior.
- There exists an optimal k that balances bias and variance; too small k increases variance, too large k increases bias.
- Computational cost scales linearly with dataset size, but can be mitigated by indexing structures such as KD-trees.

2.7 Limitations and Potential Improvements

While the method is flexible and data-driven, it can suffer from:

- Sensitivity to irrelevant features or poor distance metric choice.

- Reduced performance in high-dimensional covariate spaces (curse of dimensionality).
- Computational inefficiency for very large historical datasets.

These limitations motivate our exploration of *Topological Data Analysis* (TDA) kernels and adaptive k-selection as possible enhancements, discussed in subsequent sections.

2.8 Replication Setup

Following Lin et al. [4], we implemented the k-NN estimator using simulation data generated from a non-stationary $E_2/M/1/c$ and $H_2/M/1/c$ queueing system. The arrival rate functions $\lambda(t)$ and service rate μ were chosen to match the experimental settings in the original paper. For each simulation run, we recorded:

1. Covariate vector $\mathbf{x}(t)$ at arrival epochs.
2. Corresponding realized virtual waiting times $V(t)$.

We then applied the k-NN estimation procedure for varying values of k and computed the mean squared error (MSE) relative to the true virtual waiting times.

2.9 Results and Observations

Our replication confirms the main findings of Lin et al. [4]:

- The k-NN estimator performs well when a sufficiently large historical dataset is available.
- Performance degrades when the number of available samples is small, due to the locality of the k-NN search.
- Computational cost increases with sample size, as the nearest neighbor search must be repeated for each query.

These limitations motivate our exploration of alternative methods, such as kernel-based TDA and adaptive k-NN, to improve estimation efficiency and robustness.

3 Adaptive k-NN Estimator (Zihao Ma)

3.1 Problem and Notation

We observe arrival–wait pairs from multiple independent replications,

$$\mathcal{D} = \{(r_i, t_i, Y_i)\}_{i=1}^{N}, \qquad r_i \in \{1, \ldots, R\}. \tag{3.1}$$

Here t_i is the arrival time and Y_i is the realized (arrival-view) virtual waiting time. For any query time t, the equal–weight k-NN estimator is

$$\widehat{v}_k(t) = \frac{1}{k} \sum_{j \in \mathcal{N}_k(t)} Y_j, \tag{3.2}$$

with $\mathcal{N}_k(t)$ the indices of the k nearest neighbors in time. Near the left/right boundary we use one-sided neighborhoods to reduce edge bias, and we impose replication balancing via a soft per-replication cap to avoid a single replication dominating the average.

3.2 Why Adaptive $k(t)$?

Local curvature, noise level, and data density vary with t. A single global k cannot simultaneously optimize bias and variance across time: a smaller k is preferable where the signal is highly curved or data are plentiful (bias reduction), while a larger k is preferable where the signal is smoother, the data are sparse, or noise is higher (variance reduction). This motivates learning a *function* $k(t)$ and using the adaptive estimator $\widehat{v}_{k(t)}(t)$. Our two complementary strategies align with the per-query adaptive-k idea in Sun and Huang [7] and with the learn-to-predict-k paradigm behind A-FKNN [1].

A. Local-CV adaptive k (no training). At each anchor t_0, we form a small validation batch $S(t_0)$ drawn across replications and select k by minimizing a local leave-one-replication-out objective:

$$k^\star(t_0) \in \arg\min_{k \in \mathcal{K}(t_0)} \frac{1}{|S(t_0)|} \sum_{j \in S(t_0)} \Big(\widehat{v}_k\big(t_j;\, \mathcal{D} \setminus \mathcal{D}_{r_j}\big) - Y_j \Big)^2 \;+\; \alpha \Big(\tfrac{k}{k_{\max}(t_0)} \Big)^2, \tag{3.3}$$

where $\mathcal{D}_{r_j}$ contains all samples from replication r_j, $\mathcal{K}(t_0)$ is an (upward-biased) candidate set, and $k_{\max}(t_0)$ is a local density cap. We then median-filter $k^\star(t)$ along t to suppress jitter.

Virtual waiting time under Local-CV. Let $\mathcal{P}(t)$ be a boundary-aware candidate pool around t (one-sided near edges), and let $\mathcal{N}_{k^\star(t)}(t) \subset \mathcal{P}(t)$ be the $k^\star(t)$ nearest indices after applying a soft per-replication cap $m_{\max}$. The reported estimator is the equal-weight mean over the selected neighbors:

$$\boxed{\widehat{v}_{\mathrm{LCV}}(t) = \frac{1}{k^\star(t)} \sum_{j \in \mathcal{N}_{k^\star(t)}(t)} Y_j} \tag{3.4}$$

B. AFKNN: learn $k(t)$ with a forest (train once, apply fast). Following the spirit of Bian et al. [1], we *label* many anchors t_0 with locally optimal $k^\star(t_0)$ via a small-batch, cross-replication bias–variance objective (with a light penalty on large k):

$$\mathrm{Obj}(k; t_0) = \frac{1}{M} \sum_{j=1}^{M} \underbrace{\big(\widehat{v}_k(t_j) - v_{\mathrm{ref}}(t_j) \big)^2}_{\text{bias}^2} + \lambda \cdot \underbrace{\widehat{\mathrm{Var}}_k(t_j)}_{\text{variance}} + \alpha \Big(\tfrac{k}{k_{\max}(t_0)} \Big), \tag{3.5}$$

with a practical variance proxy

$$\widehat{\mathrm{Var}}_k(t) \;\approx\; \frac{1}{k(k-1)} \sum_{j \in \mathcal{N}_k(t)} \big(Y_j - \widehat{v}_k(t)\big)^2. \tag{3.6}$$

Here $v_{\mathrm{ref}}(t)$ is a smooth mechanistic reference used only to stabilize labels, and $k_{\max}(t_0)$ depends on the local density. We then train a random-forest regressor to predict $\log k^\star(t)$ from compact features (normalized time, a local density surrogate, local variance, left/right asymmetry, and edge flags). At inference, we predict $k(t)$ on a grid, apply density-aware lower/upper guards, median-filter $k(t)$, and finally compute the equal-weight k-NN estimate. This "learn $k(t)$" idea is related to the adaptive-k view in Sun and Huang [7].

Virtual waiting time under AFKNN. Let $f_\theta(\cdot)$ be the trained forest, $x(t)$ the feature vector at t, and define

$$k_\theta(t) \;=\; \mathrm{clip}\Big(\,\mathrm{median\_run}\,\big(\,\mathrm{round}(\exp(f_\theta(x(t))))\big),\; k_{\min}(t),\; k_{\max}(t)\Big), \tag{3.7}$$

where $k_{\min}(t)$ and $k_{\max}(t)$ are tied to effective local sample size and boundary constraints. Using the same boundary-aware pool and replication balancing as above, we estimate

$$\boxed{\widehat{v}_{\mathrm{AFKNN}}(t) = \frac{1}{k_\theta(t)} \sum_{j \in \mathcal{N}_{k_\theta(t)}(t)} Y_j} \tag{3.8}$$

3.3 Implementation Details (high level)

- **Boundary handling:** near the left (right) edge we restrict neighbors to the right (left) to reduce edge bias.
- **Replication balancing:** a soft per-replication cap during neighbor selection and local CV.
- **Global $k^\star$ (for A):** LORO-CV with the same equal-weight k-NN model used at inference.
- **AFKNN labeling (for B):** dense k-candidates, local-density upper bounds, within-anchor normalization of bias/variance terms, and light regularization to avoid saturating at large k.
- **Smoothing & guards:** running median on $k(t)$; lower bounds tied to N_{eff} and moderate upper bounds (e.g., $0.5\,N_{\mathrm{eff}}$).

3.4 Advantages over fixed k-NN

1. **Local bias–variance adaptation.** $k(t)$ automatically increases in sparse/noisy regions (variance reduction) and decreases where curvature is high (bias reduction).
2. **Boundary robustness.** One-sided neighborhoods substantially reduce boundary bias relative to blind symmetric k-NN.

3. **Replication-aware stability.** Balancing neighbors across replications prevents a single trajectory from dominating estimates.

4. **Two usage modes:**

   - *Local-CV (A):* training-free, distribution-agnostic, robust to train–test mismatch; cost is higher runtime (per-t CV) and potential jitter without smoothing.
   - *AFKNN (B):* train once, then fast inference; bias–variance regularization yields smoother $k(t)$ and better global bias control, with interpretability via feature importances and $k^\star$ diagnostics; requires a simulation/mechanistic reference that approximates the target regime.

3.5 Complexity and Practical Knobs

For (A), complexity scales with the grid size and the local CV budget (number of candidate k's and validation points per t). For (B), training (labeling + forest) is the heavy step, while inference is near-linear in grid size. Key knobs include the candidate k granularity, validation batch size, the median-filter window for $k(t)$, replication caps, and (for AFKNN) the bias–variance weights (λ, α) and local-density caps.

3.6 Summary

Both adaptive strategies share the same final estimator form but differ in how $k(t)$ is obtained: Local-CV is training-free and fully data-driven; AFKNN learns a mapping from local conditions to $k(t)$ for fast and stable application. In our experiments, both substantially outperform fixed-k baselines in regimes where curvature, noise, and density vary over time.

4 KDE-Mode Clustering Estimator (Shaozhe Ke)

In queueing systems with time-varying arrivals, the estimation of the virtual waiting time function $V(\tau_0)$ requires methods that adapt to nonstationarity and local structural patterns in the arrival process. While k-nearest neighbor (k-NN) regression provides a simple local averaging mechanism, its performance critically depends on the choice of the parameter k. In small samples, k can be tuned by cross-validation or heuristic rules, but in large-scale simulation settings, searching for the optimal k by utilizing **Algorithm 1** becomes computationally expensive and often impractical. Moreover, k-NN relies exclusively on distance in the time domain, ignoring potential multimodal structures in the underlying arrival distribution. In contrast, density-based clustering from TDA [8, 3, 6] approaches exploit the estimated probability density of arrival epochs to group arrivals into statistically coherent clusters, thereby offering a more distribution-aware and scalable estimator of virtual waiting time.

In this section, we discuss two closely related methods: the *KDE-mode clustering estimator* and the *Mean-Shift clustering estimator*. Both approaches build upon the

kernel density estimation (KDE) framework but differ in the manner in which modes are detected and clusters are formed.

4.1 Kernel Density Estimation and Its Role in Clustering

Let $\{t_i\}_{i=1}^n$ denote the observed arrival epochs in a simulation run. A nonparametric estimate of the arrival time density can be constructed using kernel density estimation (KDE):

$$\widehat{p}_h(t) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{t - t_i}{h}\right), \tag{4.1}$$

where $h > 0$ is the bandwidth parameter and $K(\cdot)$ is a kernel function, typically chosen to be Gaussian or Epanechnikov. The bandwidth controls the degree of smoothing: a large h oversmooths the density, merging modes, while a small h undersmooths, producing spurious oscillations. Classical rules of thumb such as Silverman's bandwidth [6] or cross-validation-based selection may be employed.

Modes of $\widehat{p}_h(t)$ are then identified as local maxima of the density, corresponding to regions of high concentration of arrival epochs. These modes provide natural "anchors" for clustering, as arrivals in the neighborhood of the same mode can be regarded as belonging to the same latent group.

4.2 KDE-Mode Clustering Estimator

The KDE-mode clustering method proceeds in two steps: (i) estimate the density $\widehat{p}_h(t)$ and detect its modes $\{m_1, \ldots, m_K\}$; (ii) assign each arrival t_i to the cluster associated with the nearest mode. Formally, the assignment rule is

$$m(t_i) = \arg\min_k |t_i - m_k|, \tag{4.2}$$

which induces a partition of the arrival set into disjoint clusters $\{C(m_k)\}_{k=1}^K$.

For a query epoch τ_0, the nearest mode $m^*(\tau_0)$ is identified, and the virtual waiting time is estimated by averaging the observed waiting times of arrivals in that cluster:

$$\widehat{V}_{\text{KDE-mode}}(\tau_0) = \frac{1}{|C(m^*(\tau_0))|} \sum_{i \in C(m^*(\tau_0))} v_i, \tag{4.3}$$

where w_i is the realized waiting time for arrival t_i.

This estimator effectively smooths waiting time estimates within natural "phases" of the system, as identified by the density structure. It offers significant computational savings, since once clusters are formed, estimation at new query points reduces to mode lookup and cluster averaging. However, the estimator is sensitive to the accuracy of mode detection and to the bandwidth parameter. If h is misspecified, one risks either merging distinct operational regimes of the queue or fragmenting coherent regimes into artificially small clusters.

4.3 Mean-Shift Clustering Estimator

The Mean-Shift clustering method [3, 8] generalizes KDE-mode clustering by replacing the explicit mode detection step with an iterative mode-seeking procedure. For each sample point t_i, the algorithm initializes $x^{(0)} = t_i$ and updates iteratively via the Mean-Shift rule:

$$x^{(k+1)} = \frac{\sum_{j=1}^{n} K\left(\frac{x^{(k)}-t_j}{h}\right) t_j}{\sum_{j=1}^{n} K\left(\frac{x^{(k)}-t_j}{h}\right)}. \tag{4.4}$$

This update corresponds to shifting $x^{(k)}$ toward the local mean of its neighbors, weighted by the kernel function. It can be shown that the sequence $\{x^{(k)}\}$ ascends the density $\widehat{p}_h(t)$ and converges to a local mode. Points converging to the same mode are then assigned to the same cluster.

For virtual waiting time estimation, the procedure is analogous to KDE-mode clustering: once clusters $\{C(m_k)\}_{k=1}^{K}$ are determined, the estimator is defined as

$$\widehat{V}_{\mathrm{MS}}(\tau_0) = \frac{1}{|C(m^*(\tau_0))|} \sum_{i \in C(m^*(\tau_0))} v_i, \tag{4.5}$$

where $m^*(\tau_0)$ is the mode associated with τ_0.

The advantage of Mean-Shift lies in its adaptive nature: modes need not be prespecified or directly computed from the density derivative; rather, they emerge naturally from the data through iterative updates. Moreover, the algorithm is robust to irregular multimodal structures, automatically discovering clusters of arbitrary shape.

4.4 KDE-Mode and Mean-Shift

Both methods share the same conceptual foundation but differ in implementation:

- **Computational complexity.** KDE-mode clustering requires only a density estimation and a mode-finding step, followed by a simple nearest-mode assignment. Its complexity is dominated by the KDE evaluation, typically $O(n)$ per grid point. Mean-Shift clustering, in contrast, requires iterative updates for each point, with complexity scaling as $O(n \times \text{iterations})$.

- **Assignment accuracy.** KDE-mode uses a global nearest-mode rule, which may misclassify boundary points in regions with overlapping modes. Mean-Shift provides finer assignments by tracing the density landscape, leading to more robust clustering in complex densities.

- **Bandwidth sensitivity.** Both methods depend critically on the choice of bandwidth h, but Mean-Shift is often more stable, since it accounts for density gradients rather than solely for mode locations.

- **Interpretability.** KDE-mode clustering yields crisp, easily interpretable partitions of arrival epochs into phases, making it attractive for exploratory analysis of queue dynamics. Mean-Shift, while more accurate, is computationally heavier and less transparent in its cluster formation.

4.5 Discussion and Practical Considerations

In practice, KDE-mode clustering is suitable when computational efficiency is paramount, for example in large-scale simulation experiments with millions of arrivals. Mean-Shift clustering is preferable in exploratory analyses where accuracy in detecting fine-grained structures is crucial, particularly in systems with multiple overlapping arrival streams or complex diurnal patterns.

Both approaches extend the toolbox of virtual waiting time estimation beyond distance-based methods like k-NN, offering a density-aware perspective that captures the heterogeneous temporal dynamics of arrival processes. In systems where arrival rates are inherently multimodal—such as healthcare or call center settings with daily peaks—density-based clustering provides a natural way to adaptively segment the data and deliver more accurate conditional waiting time estimates.

4.6 Summary

To summarize, density-based clustering approaches provide a principled means to leverage the estimated arrival time density for virtual waiting time estimation. KDE-mode clustering offers a simple, efficient method by assigning arrivals to pre-detected modes, while Mean-Shift clustering refines this idea by iteratively tracing density gradients. These methods complement and, in certain cases, outperform purely distance-based estimators, highlighting the importance of incorporating probabilistic structure into simulation-based statistical inference.

5 Numerical Experiments

In this section, we present a series of numerical experiments designed to evaluate the accuracy of virtual waiting time estimation under nonstationary arrivals. To this end, we first introduce the Kolmogorov forward equations (KFEs), which provide an exact numerical scheme for computing the time-dependent distribution of the queue length and thus serve as the benchmark for virtual waiting time estimation. We then compare several data-driven methods against this benchmark, including the classical k-nearest neighbor (k-NN) estimator, its adaptive variant with locally optimized k, and kernel density estimation (KDE)-based clustering. These comparisons allow us to assess both distance-based and density-based nonparametric estimators in terms of their bias, variance, and robustness to nonstationarity.

Our experiments focus on two canonical queueing models with time-varying arrivals:

- $E_2(t)/M/1/c$ **system:** an Erlang-2 arrival process with time-varying rate $\lambda(t)$, exponential service times with rate μ, a single server, and finite capacity c (including the job in service). The Erlang structure captures renewal-type variability in interarrival times beyond the Poisson case.
- $H_2(t)/M/1/c$ **system:** a hyperexponential-2 arrival process, where each arrival chooses branch 1 with probability p (rate $\lambda_1(t)$) and branch 2 with probability $1-p$

(rate $\lambda_2(t)$). This model produces higher-variance interarrival times and is useful for stress-testing estimation methods under bursty or heavy-tailed arrival patterns.

In all simulations, we specify explicit functional forms for the time-varying arrival rates (e.g., sinusoidal fluctuations), fix the service rate μ, and set system capacity c to ensure nontrivial blocking and waiting phenomena. For each configuration, KFEs are solved numerically to obtain the ground-truth virtual waiting time process, against which the performance of k-NN, adaptive k-NN, and KDE-based methods is systematically compared. All parameters are given as following table:

Table 2: Example parameter settings for simulation experiments

Experiment	Arrival Rate Function	Service Rate	Capacity
$E_2(t)$	$\lambda(t) = 2.0 + \sin(0.5t)$	$\mu = 1$	$c = 8$
$H_2(t)$	$\lambda_1(t) = 2 + 0.6\sin(0.5t)$	$\mu = 1$	$c = 8$
$H_2(t)$	$\lambda_2(t) = 0.8 + 0.4\cos(0.3t)$	$\mu = 1$	$c = 8$

5.1 Kolmogorov Forward Equations for Time-Varying Queueing Systems

To rigorously analyze the distribution of virtual waiting times under nonstationary arrivals, one of the most powerful analytic tools is the system of *Kolmogorov Forward Equations (KFEs)*. KFEs describe the transient evolution of the underlying Markov process governing the queue length and service dynamics. Unlike steady-state balance equations, KFEs capture the full time-dependent probability law, making them particularly suitable for simulation-based and transient performance analysis.

5.1.1 Model Setup

Consider a queueing system with time-varying arrivals and exponential service times. Let $\{X(t) : t \geq 0\}$ denote the state process, where $X(t)$ represents the number of jobs in the system (both in service and in queue) at time t. We assume the system capacity is finite, denoted by c, so that $X(t) \in \{0, 1, \ldots, c\}$.

Arrivals follow a nonstationary renewal process such as an Erlang-k distribution or a hyperexponential mixture, with instantaneous arrival intensity $\lambda(t)$ possibly varying over time. Service times are exponentially distributed with rate μ, and the system operates under First-Come, First-Served (FCFS) discipline.

5.1.2 Forward Equations

Define the state probability vector

$$\mathbf{p}(t) = \big(p_0(t), p_1(t), \ldots, p_c(t)\big),$$

where $p_j(t) = \Pr\{X(t) = j\}$ denotes the probability of j jobs in the system at time t.

The dynamics are governed by the Kolmogorov Forward Equations:

$$\frac{d}{dt}\mathbf{p}(t) = \mathbf{p}(t)Q(t),$$

where $Q(t)$ is the time-dependent infinitesimal generator matrix. The structure of $Q(t)$ reflects both the arrival process and the service dynamics:

$$Q(t) = \begin{bmatrix} -(\lambda(t)) & \lambda(t) & 0 & \cdots & 0 \\ \mu & -(\lambda(t)+\mu) & \lambda(t) & \cdots & 0 \\ 0 & 2\mu & -(\lambda(t)+2\mu) & \cdots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & c\mu & -c\mu \end{bmatrix}. \tag{5.1}$$

Here, each off-diagonal element captures the transition rates due to arrivals ($\lambda(t)$) or services ($j\mu$), while diagonal entries ensure that row sums equal zero. In the presence of general non-Poisson arrivals such as $E_k(t)$ or $H_2(t)$ processes, the state must be augmented with phase information, and the generator matrix $Q(t)$ expands accordingly.

5.1.3 Numerical Solution

Since $Q(t)$ depends explicitly on time, the KFEs constitute a nonautonomous system of linear ODEs. Numerical integration is required to compute $\mathbf{p}(t)$ for $t \in [0, T]$. High-order methods such as Runge–Kutta schemes or adaptive step solvers (e.g., `ode45` in MATLAB, `solve_ivp` in Python SciPy) are employed to ensure stability and accuracy.

Let $\Delta t = T/N$ be the discretization step. The system evolves according to

$$\mathbf{p}(t_{m+1}) = \mathbf{p}(t_m) + \int_{t_m}^{t_{m+1}} \mathbf{p}(s)Q(s)\,ds,$$

which is approximated numerically. This yields transient state distributions $\{\mathbf{p}(t_m)\}_{m=0}^{N}$.

5.1.4 Connection to Virtual Waiting Time

The virtual waiting time $V(\tau_0)$ can be expressed in terms of the state distribution obtained from KFEs. Specifically, for an arrival at time τ_0,

$$V(\tau_0) = \frac{1}{\mu}\Big[X(\tau_0)\Big],$$

under the FCFS discipline with exponential service. Thus, the conditional distribution of $V(\tau_0)$ is determined directly from $\mathbf{p}(\tau_0)$.

For example, if $X(\tau_0) = j$, the expected virtual waiting time is approximately j/μ. Consequently, the KFEs provide the building blocks for both estimating $V(\tau_0)$ and validating simulation-based estimators such as k-NN regression or density-based clustering.

5.2 k-NN Compared with Adaptive k-NN

Setup. We compare three estimators that share the same equal–weight k-NN smoother $\widehat{v}_k(t) = \frac{1}{k}\sum_{j\in\mathcal{N}_k(t)} Y_j$ (one–sided near the boundaries; replication balancing), but differ in how k is chosen:

(i) *fixed-k k-NN* where a global k is selected by leave-one-replication-out cross-validation (LORO-CV);

(ii) *Adaptive k-NN* where $k(t)$ is picked by local small-batch LORO-CV around each t;

(iii) *AFKNN* where $k(t)$ is predicted by a random forest trained on simulated data (seed=7 as requested) with a KFE-informed labeling objective (bias term w.r.t. KFE + a variance proxy + a light penalty on large k). Evaluation is on a common grid (600 points) for both models H2(t)/M/1/8 and E2(t)/M/1/8.

Global curves. Fig. 1 overlays the KFE "truth" with the three learned curves. Top row shows H2; bottom row shows E2. Left panels compare KFE vs. fixed-k vs. Adaptive; right panels compare KFE vs. fixed-k vs. AFKNN.

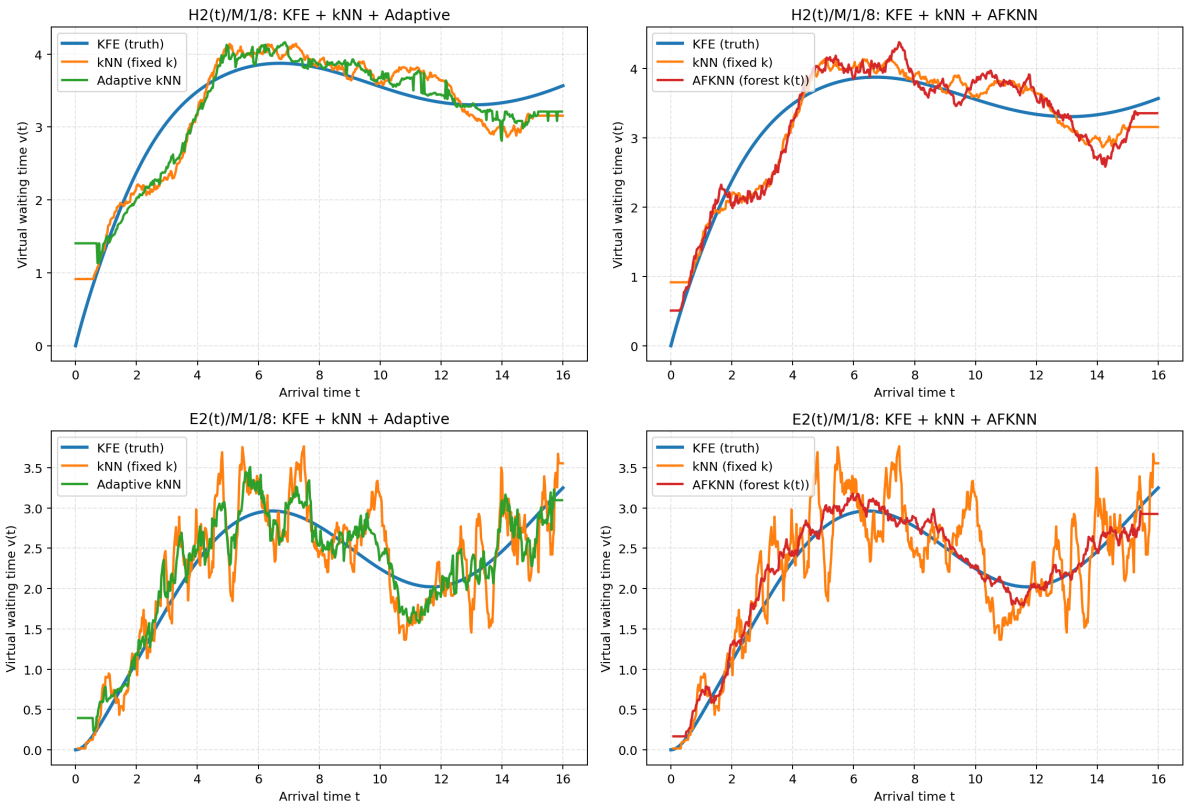


Figure 1: Global comparison on both models. Top: H2(t)/M/1/8; Bottom: E2(t)/M/1/8. Left: KFE + fixed-k + Adaptive. Right: KFE + fixed-k + AFKNN.

Pointwise error decomposition. To understand where the methods gain or lose, we plot the three pointwise curves $|\text{bias}|(t)$, $\text{Var}(t)$ and $\text{EMSE}(t) = \text{bias}^2(t) + \text{Var}(t)$. Fig. 2 gives a 2×3 panel: the first row is H2, the second row is E2; columns are |bias|, variance

and EMSE. Consistent with the global view, Adaptive k-NN tends to reduce variance in flat segments and reduce |bias| in high-curvature zones, while AFKNN delivers the smoothest and lowest-variance curve on E2.

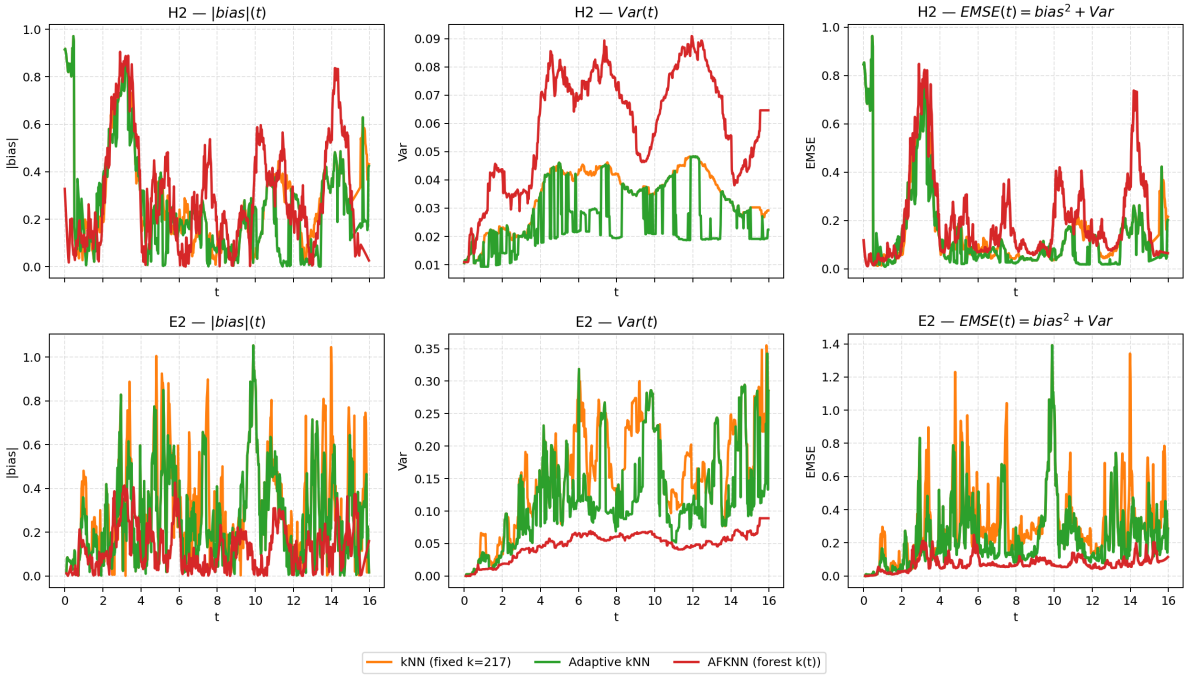


Figure 2: Pointwise error decomposition. Top row: H2; bottom row: E2. Columns: $|\text{bias}|(t)$, $\text{Var}(t)$, and $\text{EMSE}(t) = \text{bias}^2(t) + \text{Var}(t)$. Curves: fixed-$k$ (orange), Adaptive k-NN (green), AFKNN (red).

Time-averaged metrics. We also report time-averaged metrics (means over the evaluation grid).[1] Table 3 summarizes the results.

Table 3: Time-averaged metrics (lower is better).

H2 (time average)			
Method	mean |bias|	mean Var	EMSE
fixed-k	0.2839	0.0344	0.1600
Adaptive	0.2484	0.0258	0.1333
AFKNN	0.2991	0.0599	0.1995
E2 (time average)			
Method	mean |bias|	mean Var	EMSE
fixed-k	0.3163	0.1606	0.3163
Adaptive	0.2689	0.1187	0.2324
AFKNN	0.1268	0.0490	0.0750

[1] EMSE is the mean of $\text{bias}^2(t) + \text{Var}(t)$ across t.

Discussion. (1) **E2: AFKNN wins clearly.** The forest-learned $k(t)$ produces the lowest variance and the lowest EMSE (Table 3) and tracks KFE closely in Fig. 1, indicating that the KFE-informed labeling transfers well to the E2 regime.

(2) **H2: Adaptive k-NN is best among the three.** AFKNN underperforms here mainly because the forest was trained with limited seeds (as required) and a KFE-based bias term that is slightly misaligned with the H2 mixture dynamics; the learned $k(t)$ is overly smooth and variance-dominated in mid-range t, raising EMSE.

(3) **Takeaways.** When the simulation/KFE used for labeling is close to the deployment regime, AFKNN yields strong, low-variance estimates (E2). When there is a potential train–test mismatch or regime switching (H2), the training-free local-CV Adaptive k-NN is more robust and outperforms a pre-trained $k(t)$ map. In both models, using a single global k is dominated by adaptive choices across all three criteria.

5.3 k-NN Compared with KDE

In the context of estimating the virtual waiting time function $V(\tau_0)$ under time-varying arrivals, two prominent nonparametric approaches are k-nearest neighbors (k-NN) regression and kernel density estimation (KDE)-based clustering. While both methods rely on local averaging, they differ fundamentally in how "locality" is defined and how statistical information from the arrival process is utilized.

k-NN Regression. The k-NN estimator of $V(\tau_0)$ is given by

$$\widehat{V}_{k\text{-NN}}(\tau_0) = \frac{1}{k} \sum_{i \in N_k(\tau_0)} w_i,$$

where $N_k(\tau_0)$ denotes the set of k observed arrival epochs closest to τ_0 and w_i are the associated waiting times. This method is computationally simple and adapts naturally to nonstationarity through the local selection of neighbors. However, the performance crucially depends on the choice of k: too small k increases variance, while too large k introduces bias. Moreover, the optimal k generally depends on the scale of nonstationarity, which is difficult to tune in large-scale or high-intensity systems.

KDE-Based Clustering. KDE approaches, by contrast, begin by estimating the arrival density

$$\widehat{p}_h(t) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{t - t_i}{h}\right),$$

where $K(\cdot)$ is a kernel and h is the bandwidth. Clusters of arrivals are then identified via modes of $\widehat{p}_h$, and the estimator

$$\widehat{V}_{\text{KDE-mode}}(\tau_0) = \frac{1}{|C(m^*(\tau_0))|} \sum_{i \in C(m^*(\tau_0))} w_i$$

averages over the cluster $C(m^*(\tau_0))$ associated with the nearest mode $m^*(\tau_0)$. This method leverages the multimodal structure of the underlying distribution, grouping statistically similar arrivals together rather than relying solely on temporal proximity.

5.3.1 Under Different Bandwidth h

The choice of bandwidth h in kernel density estimation (KDE) plays a crucial role in determining the number and stability of detected modes. In particular, when applying KDE-mode clustering to estimate the virtual waiting time, the density landscape directly governs how arrival times are grouped into clusters. To assess this effect, we systematically vary the bandwidth parameter and visualize the resulting mode structures for both $E_2(t)/M/1/c$ and $H_2(t)/M/1/c$ systems.

For each system, we display mode construction results under eight different bandwidth values, ranging from undersmoothing (small h) to oversmoothing (large h). The left columns correspond to cases with small h, where the KDE density is highly irregular and produces many spurious local maxima, resulting in fragmented clusters. The right columns correspond to larger h, where the density surface is smoother and fewer modes are detected, sometimes merging distinct structures. These visualizations highlight the trade-off in selecting an appropriate bandwidth: smaller bandwidths capture fine details but may overfit noise, whereas larger bandwidths emphasize global structure but risk losing important local features.

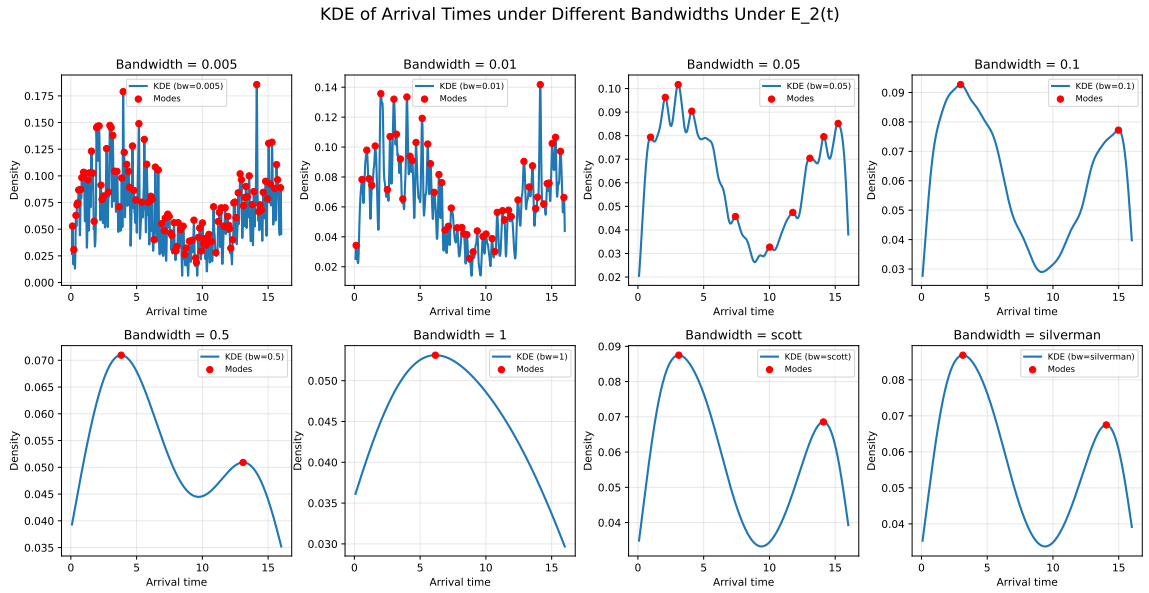


Figure 3: Mode construction in $E_2(t)/M/1/c$ under different bandwidths h. Each subfigure corresponds to a specific bandwidth, ranging from small to large.

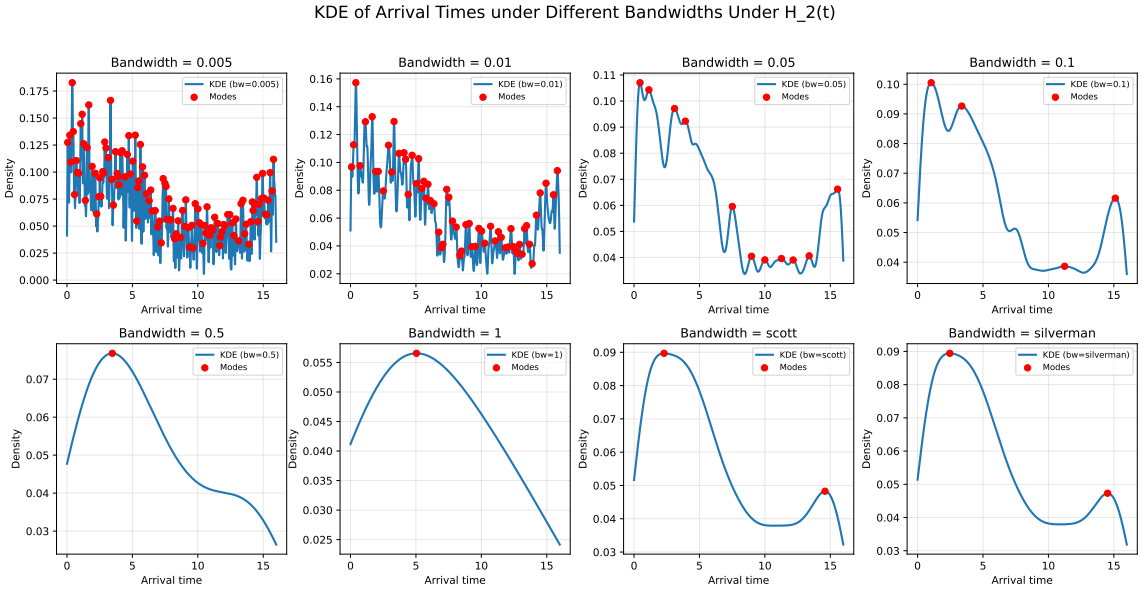


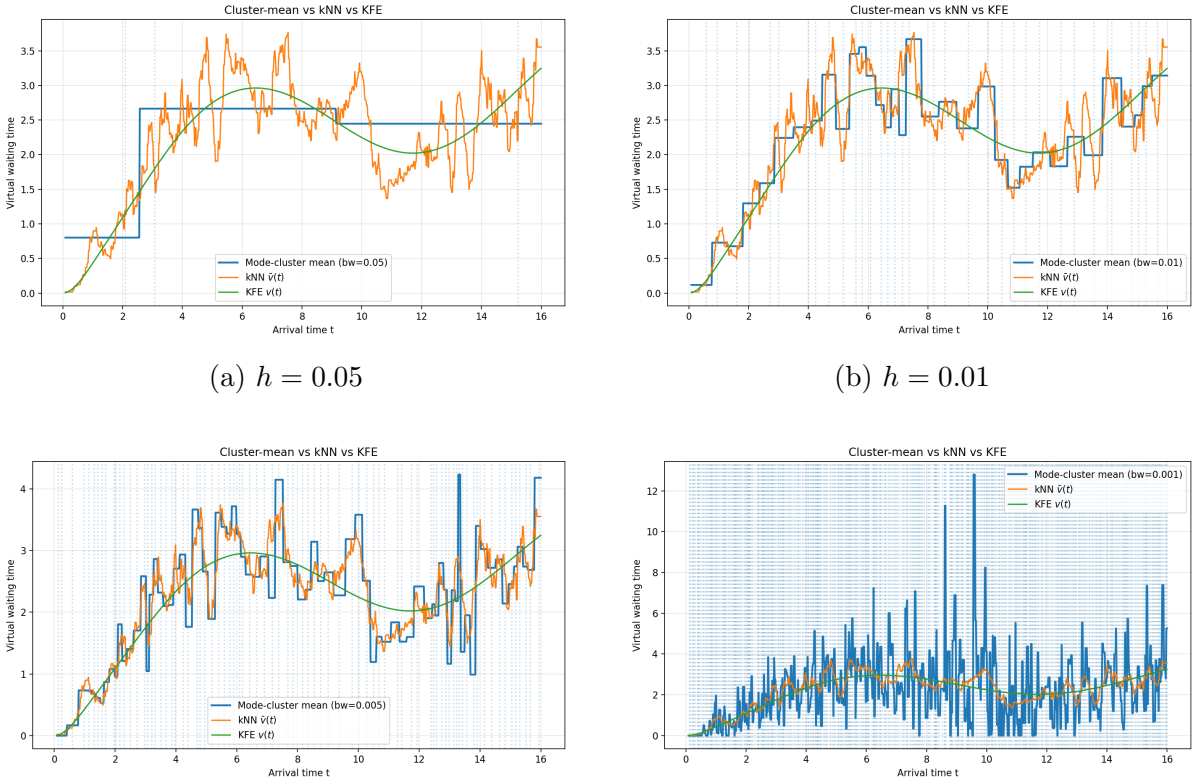
Figure 4: Mode construction in $H_2(t)/M/1/c$ under different bandwidths h.

In summary, these comparisons demonstrate that bandwidth selection fundamentally influences the mode-based clustering estimator. For $E_2(t)/M/1/c$, oversmoothing rapidly collapses the density into a small number of modes, while for $H_2(t)/M/1/c$, the mixture structure preserves multimodality across a wider range of h.

From the perspective of subsequent averaging over detected modes, it is important to preserve small-scale clusters. Hence, although smaller bandwidths may introduce additional noise, they provide higher resolution in capturing local structures. This fine granularity ensures that the averaging procedure does not overlook subtle but statistically meaningful variations in the arrival process. Therefore, in practice, bandwidth should be chosen sufficiently small to retain the essential features of the data while avoiding excessive oversmoothing.

5.3.2 Comparison of k-NN, KFES, and KDE Clustering under Different Bandwidths

We further examine how the bandwidth parameter h affects the performance of k-NN, KFES, and KDE clustering in the $E_2(t)/M/1/8$ system. In particular, we compare results under four bandwidth values: $h = 0.001, 0.005, 0.01, 0.05$. Each figure contains the outcomes of the three methods, arranged side by side for direct comparison.



(a) $h = 0.05$ (b) $h = 0.01$

(c) $h = 0.005$ (d) $h = 0.001$

Figure 5: Comparison of k-NN, KFES, and KDE clustering for the $E_2(t)/M/1/8$ system under different bandwidth values. Each panel includes three methods: k-NN (left), KFES (middle), and KDE clustering (right).

Table 4: EMSE Comparison between Mode Clustering and k-NN at $k^\star$

Bandwidth	**EMSE (Mode Cluster)**	$k^\star$	**EMSE (k-NN at $k^\star$)**
0.0001	9.9068	38	6.0605
0.001	8.1481	38	6.0605
0.005	6.2942	38	6.0605
0.010	**6.1518**	38	6.0605
0.050	6.1541	38	6.0605
0.100	6.5998	38	6.0605

From these results, we find that:

- When h is extremely small ($h = 0.001$), the clusters are highly fragmented, leading to unstable mode construction.
- With moderate bandwidths ($h = 0.005$ and $h = 0.01$), the clustering results are smoother and more interpretable. KFES in particular strikes a good balance between preserving small clusters and avoiding oversmoothing.
- For large bandwidth ($h = 0.05$), oversmoothing occurs, and fine structural details are lost.

Hence, moderate values of h (specifically 0.005 and 0.01) provide the best trade-off between accuracy and stability, and will be adopted in subsequent analyses of mode averaging. Similar results are observed for the $H_2(t)/M/1/c$ system, indicating that the conclusions are robust across different arrival processes.

6 Conclusion

In this study, we explored methods for estimating the virtual waiting time function in queueing systems with time-varying arrivals. Our investigation demonstrated that the adaptive k-NN approach improves estimation accuracy compared with the traditional k-NN method. Furthermore, we showed that kernel density estimation (KDE), when contrasted with leave-one-run-out cross-validation (LORO-CV), significantly accelerates the computation while maintaining competitive performance.

Despite these empirical advances, our work leaves several open questions. In particular, we were not able to establish rigorous theoretical guarantees for the proposed methods. Moreover, the problem of selecting an appropriate bandwidth, which plays a crucial role in mode construction and subsequent averaging, remains unresolved. These aspects warrant further investigation and constitute important directions for future research.

References

[1] Zekang Bian, Chi Man Vong, Pak Kin Wong, and Shitong Wang. Fuzzy knn method with adaptive nearest neighbors. *IEEE Transactions on Cybernetics*, 52(6):5380–5393, jun 2022. ISSN 2168-2267. doi: 10.1109/TCYB.2020.3031610.

[2] Mathieu Carrière and Andrew J. Blumberg. Multiparameter persistence images for topological machine learning. *NeurIPS Proceedings*, 2020.

[3] Keinosuke Fukunaga and Larry Hostetler. The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on Information Theory*, 21(1):32–40, 1975.

[4] Yujing Lin, Barry L. Nelson, and Linda Pei. Virtual statistics in simulation via k nearest neighbors. *INFORMS Journal on Computing*, 2019. doi: 10.1287/ijoc.2018.0839.

[5] Fengchun Qiao and Xi Peng. Topology-aware robust optimization for out-of-distribution generalization. *ICLR*, 2023.

[6] B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, 1986.

[7] Shiliang Sun and Rongqing Huang. An adaptive k-nearest neighbor algorithm. In *2010 Seventh International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2010)*, volume 1, pages 91–94, Yantai, China, 2010. IEEE. ISBN 978-1-4244-5934-6. doi: 10.1109/FSKD.2010.5569740.

[8] Larry Wasserman. Topological data analysis. *arXiv preprint arXiv:1609.08227*, 2016.