Simulation Shapelets: Comparing Characteristics of Time-Dynamic Trajectories

Graham Laidler[a], Barry L. Nelson[b], and Nicos G. Pavlidis[a]

[a]STOR-i Centre for Doctoral Training, Lancaster University, Lancaster, LA1 4YW, UK; [b]Department of Industrial Engineering & Management Sciences, Northwestern University, Evanston, IL 60208, USA

ARTICLE HISTORY
Compiled February 18, 2024

ABSTRACT
Shapelets are short, interpretable patterns in temporal data which can be characteristic of a class. In this paper, we identify shapelets from the trajectories of discrete-event simulation to indicate the characteristic dynamic behaviours of competing system alternatives. This deviates from traditional simulation output analysis, in which estimations of time-averaged performance measures overlook the more fine-grained time-dynamic features that shape the evolution of a system. We propose a shapelet methodology tailored towards simulation trajectories, and provide mathematical observations to support its implementation. To illustrate the potential of this methodology, we demonstrate its application to three examples. In particular, we reveal disruption recovery behaviour in a manufacturing simulation, provide a means for dynamic model validation, and expose the typical joint behaviour of a multivariate system state. By offering a visual characterisation of trajectories, we find that simulation shapelets can promote a deeper understanding of the dynamic behaviour and performance of simulated models.

KEYWORDS
Discrete-event simulation; trajectories; time-dynamic; shapelets; simulation analytics; deeper comparisons

1. Introduction

A stochastic simulation is a representation of a time-dynamic process. In spite of this, long-run average performance remains the prevailing mode of output analysis (Nelson & Pei, 2021). Chasing precise estimations of long-run behaviour leads to dynamic simulation outputs being averaged over both time and replications. Although an important view of simulation performance, this neglects insights into the dynamic behaviours that characterise a working system.

In this paper, we present a methodology for investigating the dynamic behaviour of stochastic simulation. We borrow the concept of shapelets (Ye & Keogh, 2011) from the literature on time series classification. Shapelets are local segments of a series that represent the characteristic patterns of behaviour by which we can discriminate among classes. Applied to trajectories of simulation variables, we find that shapelets

CONTACT Graham Laidler. Email: gllaidler@gmail.com

can identify the typical dynamic features which characterise and distinguish competing system alternatives.

We use the term *trajectory* to refer to a measurable quantity in a simulation model recorded as a continuous-time function. For example, the number-in-system in a discrete-event queueing model constitutes an integer variable that changes value only at the time of an arrival or departure event, and thus follows a piecewise constant trajectory. We specifically consider piecewise constant trajectories generated by discrete-event simulation over a finite time horizon. This affords us an enviable position with regards to data generation. We possess a ready source of clean, measurement-error-free data, with control over the form and quantity of output trajectories. Consequently, our controlled simulation environment provides advantages to a shapelet analysis which most time series applications do not. While our primary interest is in the benefits of shapelets to simulation, we are thus encouraged by reciprocal benefits of simulation to shapelets.

Our contributions in this paper are threefold. Firstly, we adapt a shapelet methodology from the setting of discretely sampled time series to the setting of piecewise constant simulation trajectories. In doing so, we offer a general-purpose methodology for output analysis. However, the computational demands of a shapelet search can be significant. Therefore, our second contribution exploits the piecewise constant structure to provide mathematical results which ease this computational aspect and simplify an implementation. Thirdly, we provide proof-of-concept illustrations of some possible applications for simulation shapelets. Specifically, we demonstrate the use of shapelets in identifying the differing dynamic response to disruption experienced by a manufacturing simulation under two design alternatives. We also show their potential for assisting operational model validation, as well as for exploring a multivariate state process. These examples demonstrate the value in moving an analysis beyond average performance, and the various benefits that shapelets can bring to an understanding of simulation behaviour.

The paper is structured as follows. Section 2 establishes our motivation and background, including an overview of time series shapelets. Section 3 describes the proposed extension to simulation shapelets, and establishes mathematical results to facilitate its implementation. We illustrate some potential applications in Section 4, showing the versatility of the approach in providing insight for various purposes. The paper concludes with a summary in Section 5.

2. Background

This section provides context for our work. (Reviewer 1, 1-3.) This may be the easiest place to mention Industry 4.0 without derailing the paper too much. Can mention specific simulation technologies, ie. digital twins, and refer to example in Section 4.2 which may be relevant to this. Section 2.1 contains a brief discussion of simulation trajectories, while Section 2.2 introduces the concept of time series shapelets, and reviews existing literature.

2.1. Simulation Trajectories

Discrete-event simulation describes a modelling paradigm in which the operation of a real system is represented by a random sequence of events occurring at discrete points in time. The state of the system can be described by a collection of variables

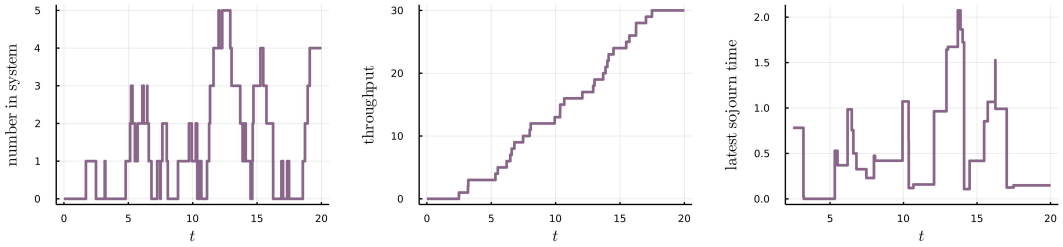

Figure 1. Examples of piecewise constant, continuous-time trajectories from a simulation of an M/M/1 queue.

which undergo instantaneous changes at event times, and otherwise remain constant. Consequently, the trajectories of simulation state variables take the form of piecewise constant functions of simulation time. We can also consider output variables such as waiting times or number of completions. Figure 1 provides some examples of trajectories from the simulation of an M/M/1 queue. From left to right, we show the number-in-system, the throughput, and the latest sojourn time as functions of simulation time.

Over the years, time series methods have been routinely applied to dynamic simulation output. However, the overarching purpose for such work has been for the study of steady-state behaviour. Fishman & Kiviat (1967) introduced a spectral analysis toolbox for simulation generated time series, with several other authors developing similar analyses to serve specific models (Heidelberger & Welch, 1983; Lada, Wilson, Steiger, & Joines, 2007). Autoregressive models have been applied to dynamic simulation outputs to assist steady-state mean and variance estimation (Schriber & Andrews, 1984; Yuan & Nelson, 1993). In fitting such models, we can imagine interest in the interpretations of the autoregressive coefficients and residual variance. However, these models rely on a weak stationarity assumption, and an autocovariance depending only on the time lag. We consider these restrictions limiting for the admission of general non-stationary trajectories.

Recently, Nelson (2016) highlighted the need and opportunity for output analysis to move towards time-dependent characterisations. This prompted the emerging field of *simulation analytics*, which has since produced dynamic metamodelling for performance predictions conditional on time or state information (Laidler et al., 2020; Lin, Nelson, & Pei, 2019). In a direct effort to characterise dynamic simulation behaviour, Fourier analysis has been successfully applied to the time series of queue length and number-in-system trajectories, with coefficient magnitudes providing interpretation with regards to system congestion (Morgan & Barton, 2022). This approach provides a statistical characterisation of dynamic behaviour. Our own approach provides a *visual* characterisation, while retaining the continuous-time form of simulation trajectories. We base our approach on a methodology appearing in time series classification, which we discuss subsequently.

2.2. Time Series Shapelets

Time series shapelets were introduced by Ye & Keogh (2011) as a fast (once trained) and interpretable means of time series classification. A shapelet refers to a short subsequence whose appearance or absence in a time series can be informative of its class. This exploits the idea that the characteristic behaviour of a class of series tends to re-

veal itself over short patterns and local variations rather than on the global structure. Furthermore, the natural visualisation of shapelets can offer valuable interpretation. Shapelets have been shown to provide useful insights and classification accuracy across a range of applications, including motion capture (Shajina & Sivakumar, 2012), health monitoring data (Zorko, Frühwirth, Goswami, Moser, & Levnajić, 2020), and detection of wind, wave, and seismic events (Arul & Kareem, 2021).

To formalise the key concepts used in time series shapelets, we introduce a time series $Z = z_{1:m}$ as a sequence of m real-valued variables, and similarly a shapelet $S = s_{1:\ell}$ of length $\ell \leq m$. The methodology relies on a notion of distance between a shapelet and a series. Initially, we define a symmetric, real-valued distance function, d, between $S = s_{1:\ell}$ and $S' = s'_{1:\ell}$, two length ℓ sequences. The squared Euclidean distance,

$$d(S, S') = \sum_{i=1}^{\ell} (s_i - s'_i)^2, \tag{1}$$

represents a common choice, although alternatives including correlation functions (Cetin, Mueen, & Calhoun, 2015) and Dynamic Time Warping (Shah, Grabocka, Schilling, Wistuba, & Schmidt-Thieme, 2016) have also been used. The distance between the shapelet, S, and the series, Z, is then defined by the closest appearance of S along the length of Z, as measured by d:

$$\operatorname{dist}(S, Z) = \min_{t \in \{1,2,\ldots,m-\ell+1\}} d(S, z_{t:t+\ell-1}).$$

With this distance function, we can assess the quality of a shapelet in providing class discrimination. We are given a dataset, D, containing the classified time series, $Z_1, Z_2, \ldots, Z_n$. Over this dataset, a shapelet, S, generates the set of distances $\{\operatorname{dist}(S, Z_1), \operatorname{dist}(S, Z_2), \ldots, \operatorname{dist}(S, Z_n)\}$. Therefore, together with a distance threshold, γ, S divides D into two subsets: $D^{\text{near}} = \{Z_i \in D \colon \operatorname{dist}(S, Z_i) \leq \gamma\}$ and $D^{\text{far}} = \{Z_i \in D \colon \operatorname{dist}(S, Z_i) > \gamma\}$. The ability of S to discriminate classes is based on how well, for some γ, the classifications in D can be separated between D^{near} and D^{far}. This is measured by information gain, which begins with a definition of the entropy of a set of classified objects. For the time series dataset D containing n_c series of class $c \in C$ and $n = \sum_{c \in C} n_c$ total series, the entropy of D is defined as

$$H(D) = -\sum_{c \in C} \frac{n_c}{n} \log\left(\frac{n_c}{n}\right).$$

After splitting D with the shapelet, S, and distance threshold, γ, this entropy becomes

$$\hat{H}(D, S, \gamma) = \frac{|D^{\text{near}}|}{n} H(D^{\text{near}}) + \frac{|D^{\text{far}}|}{n} H(D^{\text{far}}),$$

and the information gain of this splitting strategy can be written as

$$I(D, S, \gamma) = H(D) - \hat{H}(D, S, \gamma).$$

An optimal distance threshold for S on D is defined as a value, $\gamma^\star_{S,D}$, such that $I(D, S, \gamma^\star_{S,D}) \geq I(D, S, \gamma)$ for all other $\gamma \in \mathbb{R}$. The goal of a shapelet search on D is to

find a shapelet, S, that maximises $I(D, S, \gamma^{\star}_{S,D})$. While in theory there are no restrictions on the values of S, typical practice is to extract a finite pool of candidates by considering all subsequences of permissible length occurring in D. Assuming Z_i has length m_i, the permissible shapelet lengths might be considered as $\mathcal{I} = \{2, 3, \ldots, \min m_i\}$, although typically domain knowledge can be used to reduce the size of $\mathcal{I}$ and expedite the search without sacrificing meaningful shapelets.

Alternative quality measures to the information gain, such as the F-statistic (Hills, Lines, Baranauskas, Mapp, & Bagnall, 2014) and Kruskal-Wallis and Mood's Median tests (Lines & Bagnall, 2012), have also been explored, with claims of faster computation and similar discrimination. However, the information theory framework naturally lends itself to the original decision tree classifier implemented by Ye & Keogh (2011). Framing the shapelet classifier as a decision tree retains the advantages of interpretability, while readily extending the binary architecture of shapelet splitting to multi-class problems. Beyond decision trees, however, Hills et al. (2014) proposed a shapelet transformation by taking the distances from a series to each of a collection of shapelets as classification features to be used in conjunction with a range of standard classifiers.

The computational demands of an exhaustive shapelet search are unfortunately prohibitive for many problems, and much of the subsequent literature has focused on speed-up techniques. Logical computation-saving steps such as an early abandon of unfruitful distance and entropy calculations were introduced (Ye & Keogh, 2011), and although unable to improve on the worst-case complexity, bring significant speed-up in practice. Mueen, Keogh, & Young (2011) introduced a further admissible pruning technique on the candidate pool based on an application of the triangle inequality with previously cached distance computations. However, the most significant speed-ups have been achieved by pruning via heuristic approaches (for example, see He, Zhuang, Shang, & Shi (2012)), claiming the sacrifice of exactness for speed to have little impact on classification accuracy. Alternatively, Karlsson, Papapetrou, & Boström (2016) consider using randomised subsets of the training set and candidate pool in building a random forest ensemble, while Rakthanmanon & Keogh (2013) reduce computation by using piecewise aggregate approximations of the time series.

Whilst the computational challenges surrounding shapelet methodology continue to be addressed by other researchers, our interest lies in the application and benefits of shapelets for an analysis of simulation trajectories. Could mention here that our main contribution/difference from this existing shapelet literature is to consider continuous-time. (Reviewer 1, 3.)

3. Simulation Shapelets

The interpretability offered by shapelets is a valuable feature among methods for time series classification. Common alternatives such as nearest neighbour methods provide no insight into why an object is assigned to a particular class, whereas shapelets can reveal the characteristic patterns by which classes are distinguished. It is this aspect of interpretability that attracts us as we look to characterise and understand dynamic simulation behaviour, rather than a need for classification. Our ambition is for shapelets to identify the characteristic dynamics which vary across competing system designs.

Our scope is to consider simulation over a finite time horizon. Independent replications of the same system provide us with multiple trajectories of constant length,

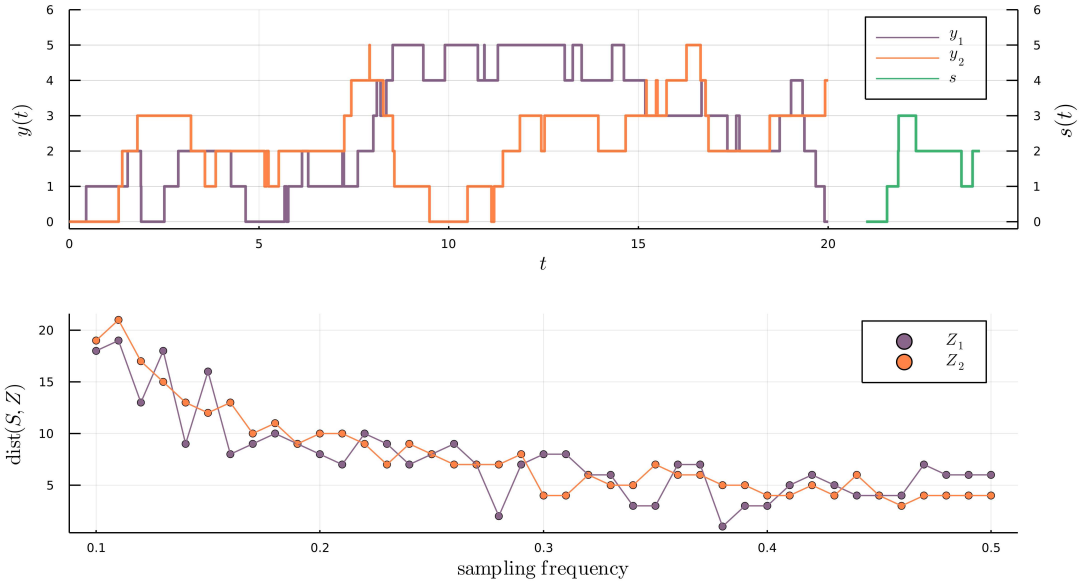

Figure 2. Top: from a capacitated M/M/1 queueing simulation, we show two continuous-time queue length trajectories, y_1 and y_2, and an example shapelet, s. We convert these into regularly sampled series, Z_1, Z_2, and S, respectively, and calculate dist(S, Z) as described in Section 2.2. Bottom: the ordering of the distance of the shapelet to each series is heavily dependent on the sampling frequency.

representing, in the context of shapelet methodology, a collection of series of one particular class. The nature of simulation can lead to replications of the same system with vastly different overall trajectories. However, we expect the underlying dynamics to consistently produce its short-term characteristic patterns, encouraging the idea that shapelets can be well-suited to an analysis of simulation trajectories. This section describes the proposed methodology and its implementation.

3.1. Continuous-time Series

The classical shapelet method accommodates time series sampled at discrete, regularly spaced time points, whereas simulation generates discrete-valued series measured in continuous time. This is the main point of difference to which we adapt.

In comparable work, continuous-time simulation trajectories have been converted into regularly sampled series to proceed with standard time series methods (Morgan & Barton, 2022). This approach introduces the decision of sampling frequency and inevitably incurs a loss of information. In a shapelet search, we also realise that results can be sensitive to this decision. For example, Figure 2 (bottom) shows the distance of a shapelet to two continuous-time trajectories from an M/M/1 queue over a range of sampling frequencies. The ordering of the series with regards to their distance to the shapelet is seen to depend heavily on the sampling frequency. Since shapelet selection relies on such distance orderings, we cannot guarantee robust results with this approach. Instead, we choose to preserve simulation trajectories in their original continuous-time format, and adapt a distance function to this context.

We denote a continuous-time simulation trajectory by $y\colon [0, T] \to \mathbb{R}$. As a piecewise constant function, we can write $t_0 = 0$, $t_m = T$, and use t_i for $i = 1, 2, \ldots, m-1$ to represent the time of the i^{th} change of y. We have $y(t) = y(t_i)$ for $t_i \leq t < t_{i+1}$. We

represent the similarly piecewise constant shapelet of length $\ell \leq T$ by $s\colon [0,\ell] \to \mathbb{R}$.

To calculate a distance between two equal-length segments, we use the L_1 distance. That is, for $s, s'\colon [0,\ell] \to \mathbb{R}$, we write

$$\|s - s'\|_1 = \int_0^\ell |s(u) - s'(u)| du. \tag{2}$$

For piecewise constant functions, this is easily calculable as a sum of rectangular areas. (Reviewer 2, 1.) Mention here that L_1 was used for visual intuition (?) and mathematical convenience, but that L_2 could also be used. Note the theorems and proofs appearing later and in Appendix can be generalised to L_p norms (result and logic of Theorem 3.3 will change, but a result should still exist). The chosen distance may depend on the intended use (eg?). We didn't try anything else, but L_1 yields simplest computation and was found to give meaningful results for simulation shapelets. Figure 3 illustrates the alteration made to the distance function as compared with the standard time series setting. Now, we define the distance between s and y as

$$\operatorname{dist}(s, y) = \min_{t \in [0, T-\ell]} \|s - y([t, t+\ell])\|_1, \tag{3}$$

where we allow shifts of the time axis to align segment domains. In particular, for $t \in [0, T-\ell]$, we define

$$\|s - y([t, t+\ell])\|_1 = \int_t^{t+\ell} |s(u-t) - y(u)| du.$$

Using the shapelet-series distance function (3), the remaining methodology surrounding the evaluation of shapelet candidates via splitting thresholds and information gain can proceed in the same way as described in Section 2.2.

In selecting candidates for the shapelet search, we recognise that the continuous-time format allows an infinite collection of shapelet candidates to be extracted from a dataset. To restrict this, we might extract candidates beginning every τ time units along our training trajectories, with the choice of τ being a consideration of computational budget as well as the granularity of the trajectories. Alternatively, to remove the need for a decision, we might extract candidates from a trajectory beginning at times at which the trajectory changes value. However, this approach may in general lead to needlessly large candidate sets. Therefore, we prefer the former approach of selecting τ, which allows us easy control over the size of the candidate set.

We also prescribe a finite selection, $\mathcal{I}$, of shapelet lengths, relying on intuition for the potential shapelet interpretations in the problem context. Our choices for τ and $\mathcal{I}$ directly control the size of the search, establishing a compromise between speed and quality. We note also that our adaptations to the continuous-time context remain compatible with existing approaches to candidate pruning (for example, see Mueen et al. (2011)).

Having outlined our methodology for continuous-time shapelets, we turn attention to its practical implementation. Calculation of the distance between a shapelet and a series (3) forms the workhorse of the method, so we continue with an observation towards its efficient computation.

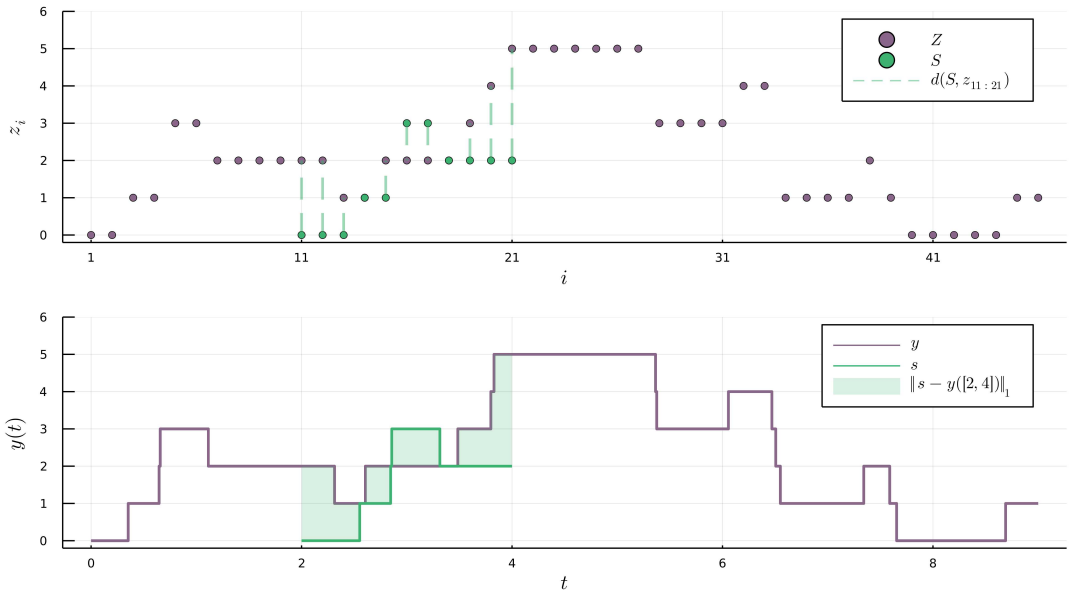

Figure 3. Illustrating the proposed alteration to the distance function in moving from the discrete-time to the continuous-time context.

3.2. Efficient Distance Computation

In calculating $\text{dist}(s, y)$, we look for the segment of y with minimum L_1 distance to s. Intuitively, we are sliding the shapelet along the series to find its best matching location, with $\|s - y([t, t+\ell])\|_1$ providing a continuous function of $t \in [0, T-\ell]$ which we seek to minimise. However, rather than requiring numerical optimisation, we notice a unique structure to this function arising from the fact that s and y represent piecewise constant trajectories. This decomposes $\|s - y([t, t+\ell])\|_1$ into piecewise linear segments in t, such that its minimum must exist at a finite collection of points. Specifically, we make use of the following result:

Theorem 3.1. *Let $y\colon [0, T] \to \mathbb{R}$ be a piecewise constant function with change times in the set $\mathcal{T} = \{0 = t_0, t_1, \ldots, t_m = T\}$ and $s\colon [0, \ell] \to \mathbb{R}$ with $\ell \leq T$ a piecewise constant function with change times in the set $\mathcal{U} = \{0 = u_0, u_1, \ldots, u_{m'} = \ell\}$. Let $U_j = \{t_i - u_j \colon i = 0, 1, \ldots, m\} \cap [0, T - \ell]$ for $j = 0, 1, \ldots, m'$, and let $\mathcal{V} = \bigcup_{j=0}^{m'} U_j$.*

Then $\|s - y([t, t+\ell])\|_1$ is linear in t for $t \in [v, v']$, where $v, v' \in \mathcal{V}$ and $(v, v') \cap \mathcal{V} = \emptyset$.

Proof. See Appendix A. □

Theorem 3.1 states that $\|s - y([t, t+\ell])\|_1$ is a piecewise linear function of t which can only change its slope at the times contained in the finite set, $\mathcal{V}$. The consequence of this for the calculation of $\text{dist}(s, y)$ is that we only need to evaluate $\|s - y([t, t+\ell])\|_1$ at points $t \in \mathcal{V}$ to find its minimum. Furthermore, calculating these distances as the sum of individual area components, as implied in Appendix A, allows for early abandon pruning as soon as the current minimum is exceeded. These observations allow substantial speed-up of the computation of $\text{dist}(s, y)$.

3.3. Location Invariance

We notice that $\|s - s'\|_1$ is heavily dependent on the vertical displacements of the segments, s and s'. In particular, two segments with identical shape but separated by a vertical shift will yield a positive distance value, while a smaller distance may be obtained between two segments which show vastly different shapes yet have similar vertical locations. In other words, similarities in shape can become swamped by the location factor. Depending on the purpose for examining simulation trajectories, however, it may be the shape rather than the location which is of greater interest in characterising the short-term system dynamics. Traditional output analysis can summarise information about the level of a series, so our primary aim with a shapelet analysis is to reveal dynamic characteristics of the shape. For this reason, we look to amend the distance function (3) such that it only reflects differences in shape.

In standard time series shapelets, z-normalisation on the individual segments is often performed prior to the distance calculation to achieve scale and offset invariance (Cetin et al., 2015). However, in our context, scale contains relevant information of the dynamics. We are comparing trajectories which all measure the same simulation quantity, such that differences in scale represent relevant dynamical differences that we need to preserve. To remove only the location factor while retaining the original shape and scale, we optimise a vertical displacement in conjunction with the horizontally sliding window. In other words, we allow a shapelet free movement vertically as well as horizontally to find its best matching location on a series. We consider the transformed shapelet $s^c\colon [0,\ell] \to \mathbb{R}$ with $c \in \mathbb{R}$ such that $s^c(t) = s(t) + c$, and define the location-invariant distance between the shapelet, s, and the series, y, to be

$$\widetilde{\mathrm{dist}}(s, y) = \min_{t\in[0,T-\ell],c\in\mathbb{R}} \|s^c - y([t, t+\ell])\|_1. \tag{4}$$

This modification ensures that the distance between a shapelet and a series depends only on resemblances to the shapelet's shape within the series. Conveniently, this contributes only a minor computational extension to the calculations implied in Section 3.2. Firstly, we retain the result of Theorem 3.1 which states that the starting position of the best matching location on the time axis can only belong to a finite set of possibilities. This arises from the following corollary:

Corollary 3.2. *Let $y\colon [0,T] \to \mathbb{R}$ and $s\colon [0,\ell] \to \mathbb{R}$ be as in Theorem 3.1. Let $c \in \mathbb{R}$ and define $s^c\colon [0,\ell] \to \mathbb{R}$ such that $s^c(t) = s(t)+c$. Let $f(t) = \min_{c\in\mathbb{R}}\|s^c - y([t,t+\ell])\|_1$.*

Then $\min_{t\in[0,T-\ell]} f(t)$ is attained by an element of the finite set $\mathcal{V}$ described in Theorem 3.1.

Proof. Assume $t^\star \notin \mathcal{V}$ minimises $f(t)$. Let $c^\star = \arg\min_{c\in\mathbb{R}}\|s^c - y([t^\star, t^\star+\ell])\|_1$. Then $\|s^{c^\star} - y([t^\star, t^\star+\ell])\|_1 \leq \|s^c - y([t, t+\ell])\|_1$ for all other $t \in [0, T-\ell]$ and $c \in \mathbb{R}$. But by Theorem 3.1, $\min_{t\in[0,T-\ell]}\|s^{c^\star} - y([t,t+\ell])\|_1$ is attained by an element of $\mathcal{V}$. This either provides a contradiction, or we have that $\|s^{c^\star} - y([t, t+\ell])\|_1 = \|s^{c^\star} - y([t^\star, t^\star+\ell])\|_1$ for some $t \in \mathcal{V}$. □

Corollary 3.2 implies that to find $\widetilde{\mathrm{dist}}(s, y)$, we only need to calculate $\|s^c - y([t, t+\ell])\|_1$ for $t \in \mathcal{V}$. Therefore, for a given point $t \in \mathcal{V}$, we now address the task of finding $\min_{c\in\mathbb{R}}\|s^c - y([t, t+\ell])\|_1$. We observe that the piecewise constant behaviour of s and y allows $\|s^c - y([t, t+\ell])\|_1$ to be expressed as a sum of rectangular areas with known dimensions. This regular structure allows us to write down its minimiser via the following theorem.

Theorem 3.3. *Let $y\colon [0,T] \to \mathbb{R}$, $s\colon [0,\ell] \to \mathbb{R}$, and $s^c\colon [0,\ell] \to \mathbb{R}$ be as in Corollary 3.2. Let $\mathcal{W}^t = \{\mathcal{T} \cap (t, t+\ell)\} \cup \{\mathcal{U} + t\}$, with ordered elements denoted by w_i^t such that $y(w_i^t) - s(w_i^t - t) \leq y(w_{i+1}^t) - s(w_{i+1}^t - t)$ for $i = 1, 2, \ldots, m_t$. Let*

$$\lambda_i = \begin{cases} \min\{w_j^t - w_i^t \colon w_j^t \in \mathcal{W}^t, w_j^t > w_i^t\} & \text{if } w_i^t \in \mathcal{W}^t \backslash \{t+\ell\}, \\ 0 & \text{if } w_i^t = t + \ell, \end{cases}$$

and let $k^\star = \min\left\{k \in \{1, \ldots, m_t\} \colon \sum_{i=1}^k \lambda_i \geq \frac{\ell}{2}\right\}$.

Then $\min_{c \in \mathbb{R}} \|s^c - y([t, t+\ell])\|_1$ *is attained by* $c^t = y(w_{k^\star}^t) - s(w_{k^\star}^t - t)$.

Proof. See Appendix B. □

In Theorem 3.3, $\{\mathcal{U} + t\}$ denotes the set $\{u_j + t \colon u_j \in \mathcal{U}\}$. Corollary 3.2 and Theorem 3.3 combine to provide an efficient computation procedure for $\widetilde{\text{dist}}(s, y)$: we evaluate $\|s^c - y([t, t+\ell])\|_1$ at the points (t, c^t) for $t \in \mathcal{V}$ and c^t as given by Theorem 3.3, with early abandoning again providing further practical speed-up.

3.4. Multivariate Shapelets

We also consider extending the application of shapelets to multivariate simulation trajectories. This would accommodate multivariate output trajectories, as well as multivariate trajectories of the system state. Changes in state variables often represent the physical movements of entities in a system, and the multivariate state can thus be highly correlated. Multivariate shapelets extracted over each dimension simultaneously may provide an insight into typical system-wide behaviours.

Bostrom & Bagnall (2017) consider options for the formulation of multivariate time series shapelets, after extracting candidates over the same indices of each dimension. To find the best matching location of a multivariate shapelet on a multivariate series, two paradigms were considered: dependent shapelets, in which the shapelet components remain in parallel, and independent shapelets, in which the components are free to move in each dimension independently. The latter approach will not necessarily reveal the common joint behaviour, and so we choose to explore the former paradigm.

We calculate a multivariate shapelet-series distance value by summing the distance contributions from each dimension. Specifically, in our continuous-time context, we calculate the location-invariant distance between a D-dimensional, length ℓ shapelet $\boldsymbol{s} = (s_1, s_2, \ldots, s_D)$ and a D-dimensional, length T series $\boldsymbol{y} = (y_1, y_2, \ldots, y_D)$ as

$$\widetilde{\text{dist}}(\boldsymbol{s}, \boldsymbol{y}) = \min_{t \in [0, T-\ell], \boldsymbol{c} \in \mathbb{R}^D} \sum_{d=1}^{D} \|s_d^{c_d} - y_d([t, t+\ell])\|_1. \tag{5}$$

The vector, $\boldsymbol{c} = (c_1, c_2, \ldots, c_D)$, contains the independent vertical displacements in each dimension. Interpreting this distance function, the best matching location of $\boldsymbol{s}$ on $\boldsymbol{y}$ is found by the shapelet components s_d moving dependently with one another across time, but independently of one another in their vertical shifts.

Calculation of $\widetilde{\text{dist}}(\boldsymbol{s}, \boldsymbol{y})$ follows straightforwardly from the univariate case, by application of the following corollary:

Corollary 3.4 (to Theorem 3.1)**.** *Let* $\boldsymbol{y}\colon [0,T] \to \mathbb{R}^D$ *and* $\boldsymbol{s}\colon [0,\ell] \to \mathbb{R}^D$ *be* D*-dimensional piecewise constant functions, with* $\ell \leq T$*. For* $d = 1, 2, \ldots, D$*, we have the univariate piecewise constant functions* $y_d\colon [0,T] \to \mathbb{R}$ *and* $s_d\colon [0,\ell] \to \mathbb{R}$*. Let* $\mathcal{V}_d$ *be the set described in Theorem 3.1 for the* d^{th} *dimension. Let* $f(t) = \min_{\boldsymbol{c}\in\mathbb{R}^D} \sum_{d=1}^{D} \|s_d^{c_d} - y_d([t, t+\ell])\|_1$.

Then $\min_{t\in[0,T-\ell]} f(t)$ *is attained by an element of the finite set* $\mathcal{V} = \bigcup_{d=1}^{D} \mathcal{V}_d$.

Proof. For each $d = 1, 2, \ldots, D$, Theorem 3.1 states that $\|s_d - y_d([t, t+\ell])\|_1$ is linear in t for t between successive elements of $\mathcal{V}_d$. Now consider successive elements $v, v' \in \mathcal{V}$ with $(v, v') \cap \mathcal{V} = \emptyset$. Since $\mathcal{V}_d \subseteq \mathcal{V}$ for all d, we must also have $(v, v') \cap \mathcal{V}_d = \emptyset$, implying that $\|s_d - y_d([t, t+\ell])\|_1$ is linear in t for $t \in [v, v']$ for all d. Therefore, $\sum_{d=1}^{D} \|s_d - y_d([t, t+\ell])\|_1$ must also be linear in t for $t \in [v, v']$. We conclude that $\sum_{d=1}^{D} \|s_d - y_d([t, t+\ell])\|_1$ is minimised by an element of $\mathcal{V}$.

Now assume $t^\star \notin \mathcal{V}$ minimises $f(t)$. Let $\boldsymbol{c}^\star = \arg\min_{\boldsymbol{c}\in\mathbb{R}^D} \sum_{d=1}^{D} \|s_d^{c_d} - y_d([t^\star, t^\star + \ell])\|_1$. Then $\sum_{d=1}^{D} \|s_d^{c_d^\star} - y_d([t^\star, t^\star + \ell])\|_1 \leq \sum_{d=1}^{D} \|s_d^{c_d} - y_d([t, t+\ell])\|_1$ for all other $t \in [0, T-\ell]$ and $\boldsymbol{c} = (c_1, c_2, \ldots, c_D) \in \mathbb{R}^D$. But we know that $\sum_{d=1}^{D} \|s_d^{c_d^\star} - y_d([t, t+\ell])\|_1$ is minimised by an element of $\mathcal{V}$. This either provides a contradiction, or we have that $\sum_{d=1}^{D} \|s_d^{c_d^\star} - y_d([t, t+\ell])\|_1 = \sum_{d=1}^{D} \|s_d^{c_d^\star} - y_d([t^\star, t^\star + \ell])\|_1$ for some $t \in \mathcal{V}$. □

By Corollary 3.4 and Theorem 3.3, we see that the calculation of $\widetilde{\text{dist}}(\boldsymbol{s}, \boldsymbol{y})$ only requires us to evaluate $\sum_{d=1}^{D} \|s_d^{c_d} - y_d([t, t+\ell])\|_1$ at points $(t, \boldsymbol{c}^t)$ where $t \in \mathcal{V}$ and $\boldsymbol{c}^t = (c_1^t, c_2^t, \ldots, c_D^t)$. Establishing a calculation of $\widetilde{\text{dist}}(\boldsymbol{s}, \boldsymbol{y})$ allows the shapelet search to proceed as in the univariate case, by calculating for each shapelet candidate the information gain achieved by the optimal splitting threshold.

In general, for variables measuring different quantities and with different scales, we might consider applying the summation (5) with unequal weightings. We note that this adjustment would not affect the result of Corollary 3.4. In the example presented in Section 4.3, however, we calculate the multivariate distance using equally weighted dimensions.

4. Applications of Simulation Shapelets

The proposed methodology offers insight into short-term simulation behaviour, which can be helpful and informative for various objectives. Some exemplar applications of simulation shapelets are illustrated in this section.

In each example, our aim is to reveal the characteristic dynamics of two alternative systems. For this purpose, we identify shapelets as follows. We use s_1 to denote a shapelet extracted from the trajectories of system 1 that maximises the information gain, with the added condition that the proportion of system 1 trajectories in the *near* subset, defined by s_1's optimal distance threshold, $\gamma^\star_{s_1,D}$, must exceed the proportion in the *far* subset. In particular, letting D denote the complete set of training trajectories, we write $D^{\text{near}} = \{y_i \in D\colon \widetilde{\text{dist}}(s_1, y_i) \leq \gamma^\star_{s_1,D}\}$ and $D^{\text{far}} = \{y_i \in D\colon \widetilde{\text{dist}}(s_1, y_i) > \gamma^\star_{s_1,D}\}$. Then, letting $D_1 \subset D$ denote the set of system 1 trajectories, we require that $|D_1 \cap D^{\text{near}}|/|D^{\text{near}}| > |D_1 \cap D^{\text{far}}|/|D^{\text{far}}|$. Without this extra condition, s_1 may instead identify characteristic behaviour of system 2 which on occasion appears in a trajectory of system 1. In the same way, we denote by s_2 an optimal shapelet which is characteristic of system 2. In the event that multiple characteristic shapelets are found that maximise the information gain, we choose the one that maximises the

margin between the two systems, which is defined as the difference between the mean distance to system 1 trajectories and the mean distance to system 2 trajectories. This is a tie-breaking strategy suggested by Ye & Keogh (2011).

In searching for the optimal shapelets, we generate candidates using the strategy of appointing τ and $\mathcal{I}$ as described in Section 3.1. However, we realise that multiple shapelets of similar lengths extracted from similar places in a trajectory will often have large overlaps, and therefore show similar ability to discriminate classes. On this basis, we see an opportunity to avoid spending computation on shapelet candidates which are unlikely to be successful. By applying the triangle inequality, Mueen et al. (2011) introduce a mathematical strategy for candidate pruning which is based on this same idea. However, in the examples here, we take a heuristic approach to candidate selection. We extract candidates of an initial length $\ell \in \mathcal{I}$ at spacings of τ time units along each trajectory in the training set, calculating for each candidate the optimal information gain, and using admissible entropy pruning as described by Ye & Keogh (2011). Whenever a new candidate, $s^\star$, improves the current best information gain, we conduct a *local* search by extracting additional candidates from the trajectory around the starting position of $s^\star$, over the full set of shapelet lengths in $\mathcal{I}$. This focuses a search to a selection of promising candidates. This approach has proved useful in our experiments, although we leave scope for further work in the problem of effective candidate selection.

Each example demonstrates the value in looking beyond a simulation's long-run average performance to its dynamic behaviour. Section 4.1 shows the effectiveness of shapelets in targeting a comparison of a manufacturing system's dynamic response to disruption under two design alternatives. In this example, while the long-run average behaviour of the target performance indicator is similar across the two alternatives, shapelets reveal that the observed behaviour while operating is significantly different. Section 4.2 uses a simple tandem queueing simulation to demonstrate the potential for shapelets to assist operational model validation. Assuming the availability of trajectories from a real-world system, we look for shapelets to assess the capability of two simulation models in mimicking the real system's dynamic behaviour. Using the same simulation, Section 4.3 explores the possibility for multivariate shapelets to provide insights into the joint behaviour of system state trajectories. Again, we find a deeper comparison and understanding of system behaviour provided by the dynamic characteristics.

The simulation models behind these examples were implemented using Simul8 software (ref). In running replications of competing systems, we use the established practice of common random numbers (Nelson & Pei, 2021), or coupling (ref), which ensures that replications of each system experience similar random input behaviour. This highlights an aspect of simulation control which can be helpful towards a shapelet analysis. By minimising the differences of input randomness affecting the trajectories of each system, we expect the characteristic differences that shapelets identify to more reliably reflect the underlying dynamics of system behaviours.

(Reviewer 1, 5.) The computational complexity of the shapelet search depends on several factors, including the number of training trajectories, their length, T, the shapelet length, ℓ, and the parameters, τ and $|\mathcal{I}|$. In the multivariate case, the dimension, d, also becomes relevant. These values are chosen to allow a manageable search time, while still discovering meaningful shapelets. We report the values used in each example, and provide an indication of the search times when performed using a 1.6 GHz Intel Core i5 processor.

The simulation data and code used for the analyses and presentation of figures in

this section is made available through a GitHub repository (Laidler, 2023).

4.1. Dynamic Response to System Disruption

Systems in the supply chain and manufacturing sectors often experience disruption as a result of unexpected shortages or breakdowns of unreliable components. A system's resilience and recovery from such events is an important consideration, and reactive decisions and schedules are often sought to mitigate their impact (Mehta & Uzsoy, 1999). While traditional summary measures may capture long-term impacts, we expect the practical repercussions of a disruption to appear in a system's temporary dynamic behaviour. In this example, we look at the dynamic throughput sequence of a manufacturing system, and exploit shapelets to reveal the short-term impacts of machine failures.

Kayton, Teyner, Schwartz, & Uzsoy (1996) introduced a simplified model of a wafer fabrication facility (fab), in which semiconductor wafers undergo various processing steps at a number of stations. Their simulation experiment explored the effects of machine breakdowns and dispatching rules on time-averaged performance measures such as the average work-in-process and the bottleneck utilisation. Dispatching rules refer to the rules for priority ordering by which queueing jobs are sequenced to work centers. Borrowing the wafer fab model of Kayton et al. (1996), our intention is for shapelets to compare the impact of machine failures on the system's dynamic throughput under different dispatching rules.

We consider the rules *Least Remaining Work* (LRW) and *Most Remaining Work* (MRW), which prioritise wafers at each station according to the number of their remaining processing steps. LRW is commonly used to accelerate throughput and system de-congestion, while MRW can target specific performance goals such as minimising the maximum lateness (Kaban, Othman, & Rohmah, 2012). We simulated week-long replications featuring machine breakdowns, with the LRW and MRW dispatching rules yielding 95% confidence intervals for the weekly average throughput of $[84.1, 84.4]$ and $[82.1, 82.9]$, respectively. The similarity of these long-run statistics suggests we may find value in a more detailed dynamic comparison.

In the original work of Kayton et al. (1996), a machine which is visited once and early in the processing sequence of each wafer is suggested to have low reliability. To specifically uncover the breakdown and recovery behaviour, we consider imposing a fixed structure to the breakdowns of this machine. Specifically, we inject a breakdown lasting for a fixed duration of 16 hours at a set time in each replication. This promotes consistency among the trajectories; it ensures that each trajectory displays a breakdown and recovery period of similar proportions and at a similar time. In this way, we can isolate the breakdown effect and focus the shapelet search accordingly. This demonstrates an advantage that the simulation context provides. To uncover specific behaviour surrounding a known system event, our control over the simulation translates to control over the trajectories and allows us to manipulate a simulation and shapelet search to target the desired insights.

We take 100 training replications of the system under each dispatching rule. To generate the candidate pool for a shapelet search, the consistent structure surrounding breakdown times allows us to shrink the window for the start times of candidate shapelets. We extract candidates every $\tau = 2$ hours from within a suitable range over which the breakdown effects begin to show in each replication. To allow the shapelets to display breakdown and recovery behaviour considering a breakdown length of 16

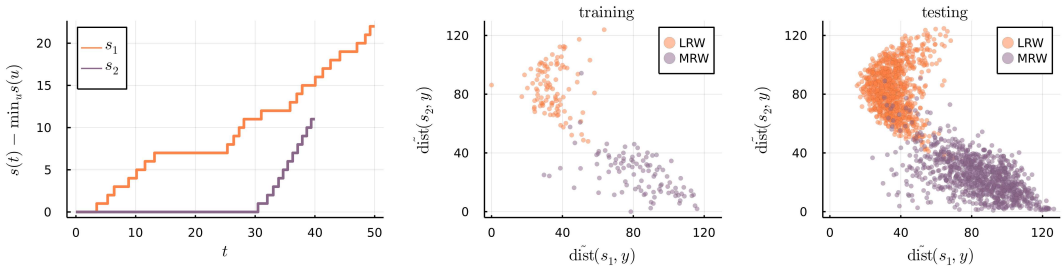

Figure 4. Comparing the impact of machine breakdowns on throughput behaviour for systems operating under the LRW and MRW dispatching rules. The characteristic shapelets are shown on the left. The scatter plots show the distribution of the 200 training trajectories (middle) and 2000 testing trajectories (right) with regards to their distances from the two shapelets.

hours, we search for shapelets with lengths of $\mathcal{I} = \{40, 45, 50\}$ (hours). Each shapelet search lasts approximately 30 minutes.

The optimal shapelets discovered are shown in Figure 4 (left). Since the shapelets are location-invariant (see Section 3.3), we in fact plot $s(t) - \min_u s(u)$, which ensures that s_1 and s_2 can easily display on the same axes. The shapelets suggest distinctly different dynamic behaviour resulting from machine breakdowns. The shapelet, s_1, which is characteristic of systems operating the LRW rule, shows relatively brief disruptions to the throughput sequence. In contrast, s_2 represents the system under the MRW rule and shows a lengthy period without any throughput. However, once s_2 resumes throughput, its rate is very quickly recovered. We realise that the MRW rule restrains throughput, with the volume of jobs nearing completion accumulating until the completion process can resume. This results in a longer period of throughput starvation following system disruption, whereas the LRW rule prioritises re-establishing the throughput at the expense of a more gradual recovery of its rate.

The scatter plots show the distance of each trajectory to each shapelet, and reveal strong separation. A set of 1000 unseen test replications of each system were used for the right hand scatter plot. These follow the same behaviour as the training replications, giving us confidence that the discovered shapelets are reliably characteristic of the two dispatching rules and not merely a result of overfitting to the training trajectories.

As stated, targeting specific behaviour allows us to impose structure to the simulation and focus our shapelet search accordingly. To test the power of shapelets without such structure, perhaps to uncover insights in the absence of prior knowledge or expectations, we repeat the experiment with random breakdown behaviour. We use distributions for the mean time to failure and mean time to repair as implied by Kayton et al. (1996), such that that replications contain breakdowns at random times and of random durations. The average breakdown duration is now 30 hours as opposed to a fixed 16 hours. Some replications contain no breakdown. However, we retain some consistency across the corresponding replications of each dispatching rule by simulating with common random numbers. As such, the differences across the dispatching rules remain in the impact and recovery of breakdowns rather than in the breakdowns themselves.

Figure 5 shows the results from the unstructured simulation. Without the computational benefits of a targeted shapelet search, we use $\tau = 10$ and $\mathcal{I} = \{70, 75, 80, 85, 90\}$ (hours), each search lasting approximately 60 minutes. Despite the inconsistent breakdown behaviour across replications, the shapelets still manage to separate the systems

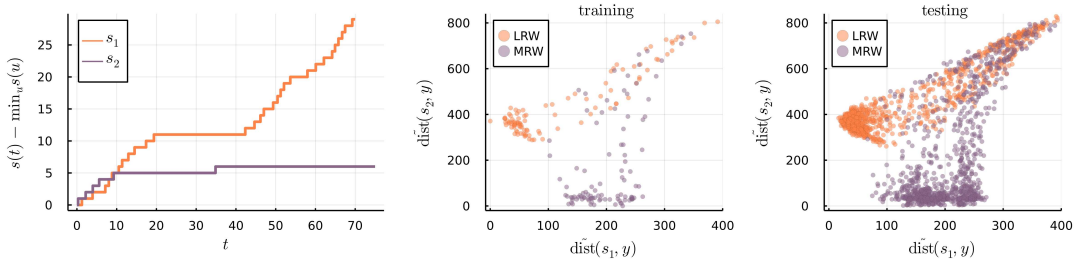

Figure 5. Repeating the results shown in Figure 4 using trajectories in which breakdowns occur randomly and with random recovery durations.

by their dispatching rule. Breakdowns under the MRW rule are again seen to produce the most disruption to throughput. The scatter plots show a number of replications with large distances to each shapelet, mainly identifying the replications in which a complete breakdown and recovery period was not observed. Revealing the characteristic behaviour surrounding breakdown impact from replications without any targeted structure gives us confidence in the ability of shapelet methodology to discover hidden and unexpected insights into dynamic simulation behaviour.

4.2. Dynamic Model Validation

Operational validation is an important aspect of simulation modelling. For effective decisions to be taken in a real system based on the output of a simulation model, it is important for these outputs to bear sufficient resemblance to those of the real system. Operational model validation often amounts to a statistical or graphical comparison between the distributional properties of output variables, such as comparing histograms of waiting times, or conducting hypothesis tests around their means (Sargent, Goldsman, & Yaacoub, 2016). Thus, model validation is often restricted to the level of output summaries and grouped behaviours.

Shapelets may provide an additional aspect to model validation by discerning differences in the dynamic behaviour of simulation trajectories. In a related context, the emergence of digital twins has increased interest in bringing model validation into an online setting. Digital twins are models which aim to synchronise with a live system, thus providing support for real-time monitoring and control. As such, validating digital twins requires an online validation of the model behaviour in comparison to dynamic data received from the real system. Hua, Lazarova-Molnar, & Francis (2022) provide an overview of opportunities and challenges in the validation of digital twins. Crucially, interest is growing in performing model validation via dynamic behaviour, and this is an area to which shapelets may be applicable.

To construct an example, we consider the three-station tandem queue depicted in Figure 6. Station I contains s_I identical servers, and we set $s_I = 1$ for $I \in \{A, B, C\}$. Each station admits infinite queueing space and generates exponential service times with rate $\mu = 0.6$. We simulate customer arrivals following a Poisson process, and let the *true* system have the smooth arrival rate function, $\lambda(t) = (\sin(2\pi t/10) + 1)/2$. In a real system, $\lambda(t)$ would be unknown, and therefore might be modelled using a piecewise constant rate function estimated from data. Two options for modelling $\lambda(t)$ with piecewise constant segments are chosen, and shown in Figure 6. Model 1 uses only two constant levels, rounding $\lambda(t)$ to the nearest integer, while model 2 uses

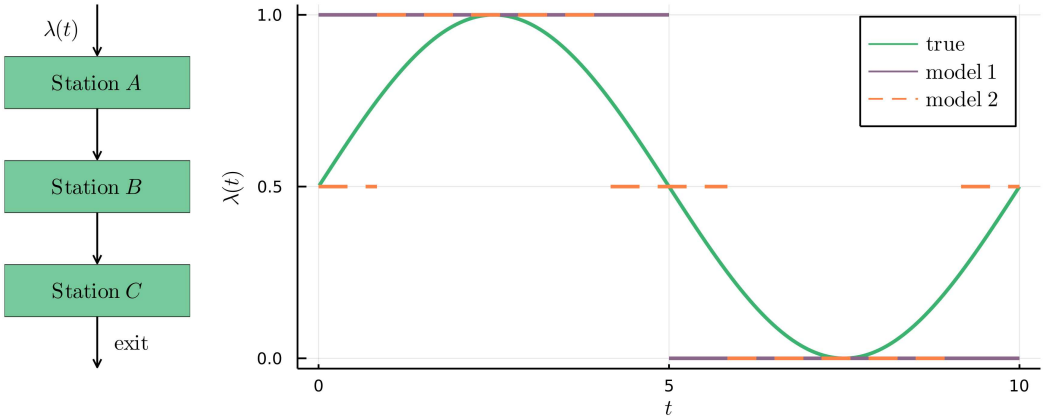

Figure 6. Left: a diagram of a tandem queue. Right: the arrival rates, $\lambda(t)$, of the true system and the two models. These repeat with period $t = 10$.

three levels, rounding to the nearest 0.5.

We consider model validation for the dynamic number-in-system behaviour, using replications of length $T = 60$ minutes after discarding a warm-up period of length 20 minutes. From 10,000 replications, the mean number-in-system for the true system is found to be 8.28. Model 1 gives a 95% confidence interval for the mean number-in-system to be $[8.38, 8.52]$, while for model 2 this confidence interval is $[8.24, 8.39]$. Based on this long-run behaviour, both models would be deemed operationally valid to within 0.25 customers. We look for a shapelet analysis to go beyond this and discern if there exist differences in the dynamic behaviour. Intuitively, we should struggle to find characteristic dynamics that distinguish a good model from the true system.

Figure 7 compares model 1 with the true system. We took 200 training replications of both the model and the true system, and searched for shapelets from the number-in-system trajectories using $\tau = 2$ and $\mathcal{I} = \mathbb{Z} \cap [8, 12]$ (minutes), each search lasting around 2.5 hours. The optimal shapelets are shown in the top left of Figure 7. The shapelet, s_1, is characteristic of the true system, and depicts a stable behaviour with arrivals and exits both occurring intermittently. On the other hand, s_2 is characteristic of model 1, and shows a long period of no arrivals followed by a sudden burst. The scatter plot in the bottom left of Figure 7 shows the distances of the trajectories from 5000 test replications of both the true system and the model to each of the shapelets. We see significant overlap, yet definite inclinations of the true system and the model to display behaviours closer to s_1 and s_2, respectively.

Fitting Gaussian distributions to the clouds of points in the test scatter plot results in the contours shown in the bottom right of Figure 7. To quantify the results of the shapelet analysis for model validation, we might consider the classification accuracy. An ideal model may generate indistinguishable behaviour from the true system and yield a classification accuracy of 0.5. Classifying the test trajectories based on their distances to s_1 and s_2 and the probability densities of the fitted distributions yielded a classification accuracy of 0.61.

Figure 8 shows the equivalent plots comparing model 2 with the true system. The discovered shapelets are less distinct from one another, while the scatter plot of test replications does not clearly show a disposition of either system towards either shapelet. The greater separation in the training scatter plot as compared to the testing

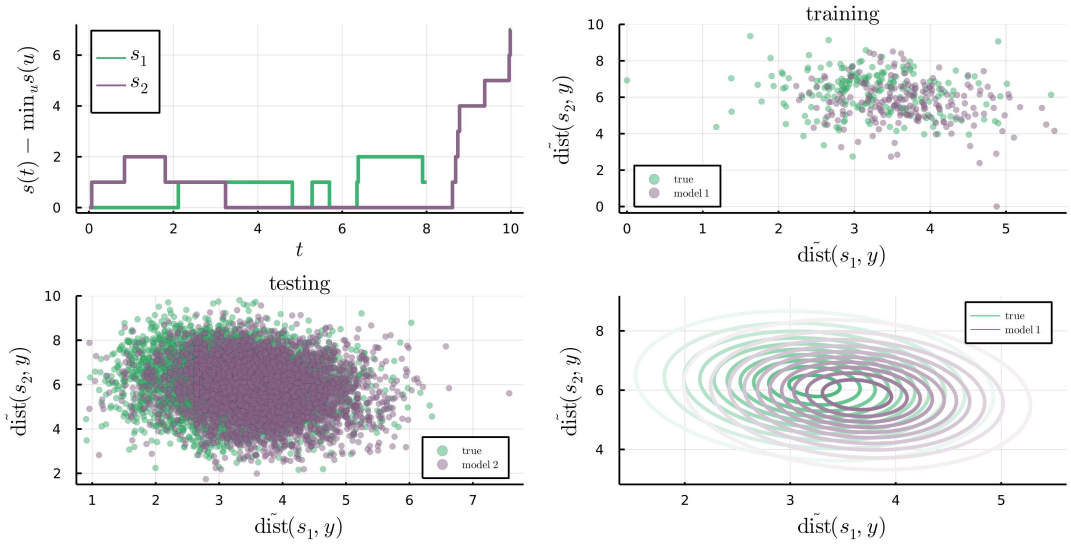

Figure 7. Comparing the dynamic number-in-system behaviour of the true system with model 1. The characteristic shapelets are shown in the top left. Gaussian distributions (bottom right) are fitted to the collection of testing trajectories (bottom left).

scatter plot suggests that the discovered shapelets may result from overfitting to the training trajectories rather than being more broadly characteristic of the two systems. The classification accuracy was found to be 0.51.

In summary, our shapelet analysis identified different characteristic behaviour of the number-in-system trajectories between the true system and model 1, but not between the true system and model 2. We conclude that model 2 is a more valid model of the true system. Referring to the model approximations shown in Figure 6, this is the conclusion we should expect. Shapelets appear able to present a comparison between systems' dynamic behaviours, and may represent a viable avenue for online model validation.

4.3. Multivariate Shapelets of the System State

To test the potential for multivariate shapelets in simulation, we remain with the tandem queueing model used in Section 4.2. We simulate Poisson arrivals with the true arrival rate function, $\lambda(t)$, seen in Figure 6, and exponential service times with rate $\mu = 0.6$ at each station. Suppose that we now have a total capacity for four servers, and consider two system alternatives:

(1) system 1: $s_A = 1, s_B = 2, s_C = 1$,
(2) system 2: $s_A = 1, s_B = 1, s_C = 2$.

We again use replications of length $T = 60$ minutes after discarding a warm-up period of length 20 minutes. Both systems yield an overall mean number-in-system of 2.29. The number-in-system trajectories themselves also show similar behaviour, and so we go further by breaking this down into the separate stations. We consider a three-dimensional system state $\boldsymbol{y}(t) = (y_A(t), y_B(t), y_C(t))$, where $y_I(t)$ represents the number of customers in station I at time t (queueing and in service).

Initially, we perform a univariate shapelet search in the three dimensions of $\boldsymbol{y}(t)$ individually. We use 100 training replications of each system, and search for shapelets

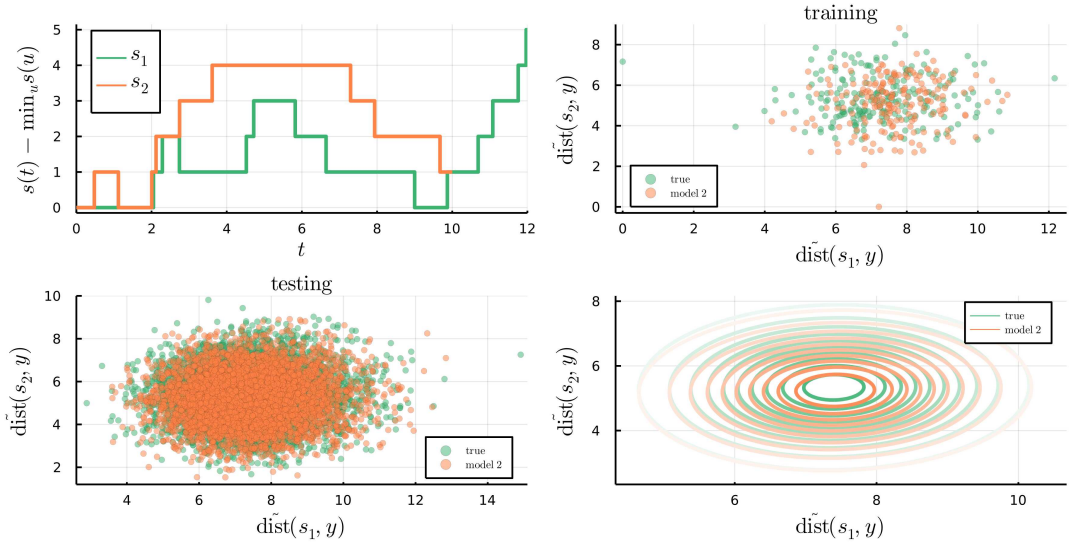

Figure 8. Shapelets are not able to discriminate between the dynamic number-in-system behaviours of the true system and model 2.

using $\tau = 2$ and $\mathcal{I} = \{10, 11, 12\}$ (minutes). Each search lasts around 30 minutes. The resulting shapelets are displayed in Figure 9, where s_{Ij} denotes the shapelet in the dimension y_I that is characteristic of system j. Common random numbers ensured identical trajectories of $y_A(t)$ across the replications of both systems, causing a shapelet search in this dimension to be futile. The differences emerge in the trajectories of $y_B(t)$ and $y_C(t)$, and as we might expect, system 1 is characterised by greater stability of y_B and volatility of y_C, while the reverse is true of system 2. The bottleneck at station C in system 1 appears more significant than the bottleneck at station B in system 2. The scatter plots show the distributions of 1000 test trajectories of each system with respect to their distances from s_{I1} and s_{I2}. The ability to discriminate the systems increases the further downstream we move.

We also perform a multivariate shapelet search, using the summation distance function (5). For this, we use $\tau = 10$, with each search lasting around 60 minutes. The results are shown in Figure 10, where $\boldsymbol{s}_1$ is characteristic of system 1 behaviour, and $\boldsymbol{s}_2$ of system 2. Retaining the multidimensional shapelet shows the dimensions moving concurrently. For example, we see that as y_A decreases, y_B increases, and as y_B decreases, y_C increases, reflecting the movement of customers through the system. We receive similar interpretations regarding the shifting of the bottleneck station between the two systems. Further, the multivariate shapelets support the conclusion of a more severe bottleneck at station C in system 1 than at station B in system 2. This may be understood from the differing server capacities at their preceding stations. At times when $\lambda(t)$ exceeds the service rate of $\mu = 0.6$, the departure rate from these preceding stations will be governed by their number of servers. As such, we expect a greater workload on station C in system 1 than on station B in system 2. The scatter plot of test trajectories shows good separation of the two systems.

In the simple example presented here, the known server allocations represent the obvious difference between systems 1 and 2, and the multivariate shapelets identify the behaviours that match our expectations for the dynamic effect of this difference on the station populations. However, the differences among systems may not always be so clear, and the results in this simple case give us confidence that a multivariate shapelet

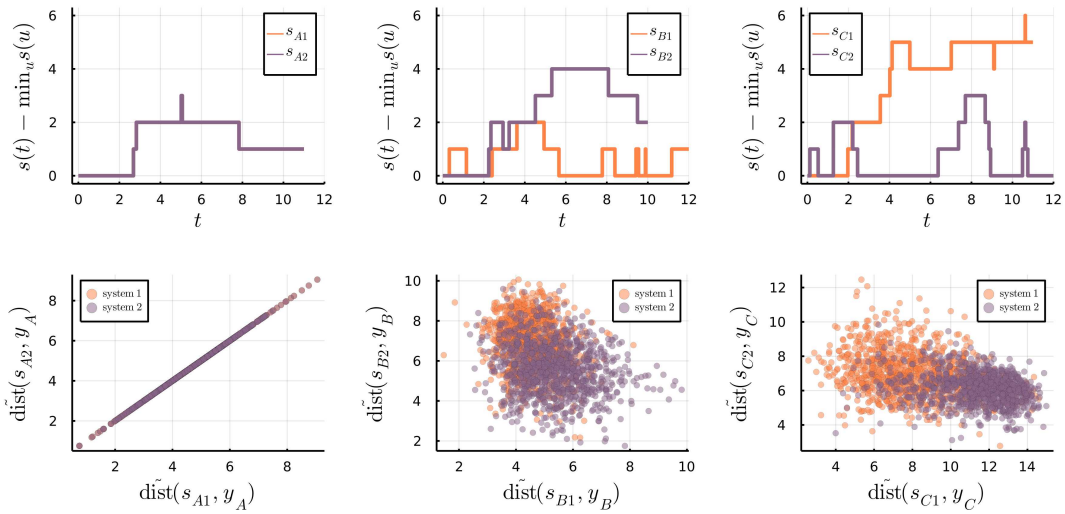

Figure 9. Independently finding univariate shapelets in the three dimensions of $\boldsymbol{y}(t)$.

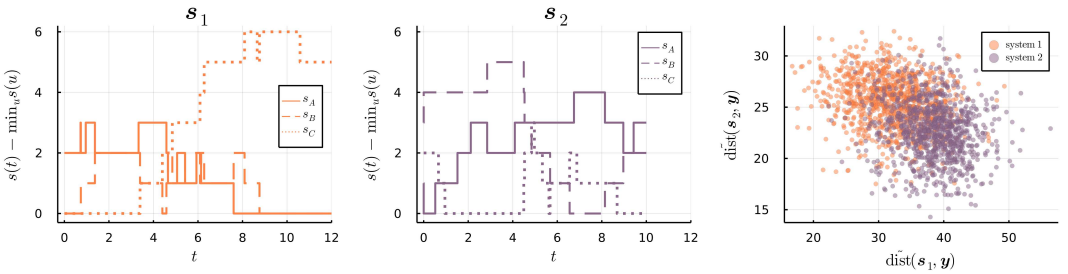

Figure 10. Finding multivariate shapelets reveals the typical joint behaviour of the station capacities for each system.

search can extract meaningful and unanticipated behaviour when competing systems feature more nuanced differences. In summary, we see promise in the application of multivariate shapelets to a simulation state process, and potential for developing a more tailored methodology for this purpose.

5. Discussion and Conclusion

Reviewer 1, 8. suggests a new section prior to the conclusion, but maybe we can just expand this section. I'm not really sure what to expand it with though. Just reiterate our place within simulation analytics? Could address Reviewer 2, 3. comment when mentioning example 4.1: "Could also be used to visualise the behaviour of existing systems, when sufficient trace data are available." Maybe borrow something from thesis conclusions, although this is mainly about further work:

"An alternative application which we have not shown is to provide a comparison between 'good' and 'bad' trajectories of the same system, as defined by some overall performance value. Replications of a system can often exhibit vastly different performance, and shapelets may be able to reveal the characteristic behaviours which distinguish this phenomenon. This might assist an understanding of the inherent variability of a system, and the conditions and short-term patterns of behaviour that define its performance.

Continuing on the side of application, the use of shapelets for dynamic model validation, as illustrated in Section 5.4.2, would benefit from a more rigorous framework. Quantifying the validity of a model based on classification accuracy suggests that we should make an effort to construct the most accurate classifier possible. With this aim, the idea of performing a shapelet transform and applying alternative classifiers on a collection of distance features, as proposed by Hills et al. (2014), seems sensible.

To understand the contribution of shapelets in the context of existing methodology, we might also explore a comparison with previous work. Simulation shapelets target a visual interpretation of dynamic behaviour for which we have not found directly comparable alternatives. However, simulation trajectories are analysed by Morgan and Barton (2022) from a statistical perspective of their Fourier transformations, and a comparison of interpretations with this approach may be possible and useful.

In terms of methodology, extensions to time series shapelets have progressed in a number of directions, which were outlined in Section 2.2. We can imagine the progression of simulation shapelets in similar directions. Importantly, we note that our main motivation for simulation shapelets is towards interpretation rather than classification. For this reason, we do not require a decision tree classifier, which makes the use of the information gain quality measure non-essential. Computing the optimal information gain requires its evaluation over n-1 possible splitting thresholds. Speed-up in the shapelet search could therefore be attained by using an alternative measure which is independent of a splitting threshold, such as the F-statistic suggested by Hills et al. (2014)."

We present a methodology for simulation output analysis which places focus on local characteristics of the underlying system dynamics. This methodology is based on the use of shapelets, which describe locally interpretable patterns capable of discriminating among time series classes. Applied to simulation trajectories, shapelets provide a means of characterising dynamic behaviour and performing deeper comparisons across competing system alternatives. Comparison across systems represents a common objective in the use of simulation, and when alternatives exhibit similar

long-run performance, we conceive value in making deeper comparisons on the basis of dynamic behaviour.

To propose a shapelet methodology adapted to the structure of simulation trajectories, we present a reframing of the original discrete-time setting to the context of continuous-time functions. We also propose a natural approach to achieving location invariance, such that we solely represent the local shape dynamics appearing in the trajectories. These adaptations introduce theory through which the piecewise constant construction of simulation trajectories lends convenient structure to the shapelet-series distance function and relieves the computational aspect of its minimisation. Through this, we establish an efficient and practical methodology for piecewise constant shapelet discovery, and demonstrate its promise in application to simulation problems. For example, we see effective results in uncovering dynamic behaviour of interest, alongside potential applications to dynamic model validation and furthering an understanding of the dynamic behaviour of a multivariate system state.

Although the computational complexity of a shapelet search poses a challenge for many real-scale problems, the progress and focus of ongoing research in addressing this aspect gives practical encouragement to the proof-of-concept provided here. Additionally, the control over data generation afforded to us by simulation should not be ignored. Targeting specific behaviour and employing control procedures such as common random numbers allow us to focus a search and improve our training set to the task of meaningful shapelet extractions.

Simulation trajectories provide a rich environment for analysis, and a shapelet-based methodology shows potential to provide valuable insight into the often overlooked dynamics of system behaviour. An increasing demand for machine learning solutions and data-driven insights across system operations gives us confidence in the scope and impact of this approach.

Disclosure statement

The authors report that there are no competing interests to declare.

Funding

This work was supported by the EPSRC funded STOR-i Centre for Doctoral Training at Lancaster University under Grant number EP/L015692/1. In addition, Barry Nelson's work was supported by the National Science Foundation of the United States under Grant number DMS-1854562.

References

Arul, M., & Kareem, A. (2021). Applications of Shapelet Transform to Time Series Classification of Earthquake, Wind and Wave Data. *Engineering Structures*, *228*, 111564.

Bostrom, A., & Bagnall, A. (2015). Binary Shapelet Transform for Multiclass Time Series Classification. *Big Data Analytics and Knowledge Discovery, DaWaK 2015*, *9263*, 257–269.

Bostrom, A., & Bagnall, A. (2017). A Shapelet Transform for Multivariate Time Series Classification. https://doi.org/10.48550/arXiv.1712.06428

Cetin, M. S., Mueen, A., & Calhoun, V. D. (2015). Shapelet Ensemble for Multi-Dimensional

Time Series. In S. Venkatasubramanian & J. Ye (Eds.), *Proceedings of the 2015 SIAM International Conference on Data Mining* (pp. 307–315). SIAM.

Fishman, G. S., & Kiviat, P. J. (1967). The Analysis of Simulation-Generated Time Series. *Management Science*, *13*(7), 525–557.

Glynn, P. W. (1989). A GSMP Formalism for Discrete Event Systems. *Proceedings of the IEEE*, *77*(1), 14–23.

He, Q., Zhuang, F., Shang, T., & Shi, Z. (2012). Fast Time Series Classification Based on Infrequent Shapelets. *2012 11th International Conference on Machine Learning and Applications*, *1*, 215–219.

Heidelberger, P., & Welch, P. D. (1983). Simulation Run Length Control in the Presence of an Initial Transient. *Operations Research*, *31*(6), 1109–1144.

Hills, J., Lines, J., Baranauskas, E., Mapp, J., & Bagnall, A. (2014). Classification of Time Series by Shapelet Transformation. *Data Mining and Knowledge Discovery*, *28*(4), 851–881.

Hua, E. Y., Lazarova-Molnar, S., & Francis, D. P. (2022). Validation of Digital Twins: Challenges and Opportunities. In B. Feng, G. Pedrielli, Y. Peng, S. Shashaani, E. Song, C. G. Corlu, L. H. Lee, E. P. Chew, T. Roeder, & P. Lendermann (Eds.), *Proceedings of the 2022 Winter Simulation Conference* (pp. 2900–2911). IEEE.

Kaban, A. K., Othman, Z., & Rohmah, D. S. (2012). Comparison of Dispatching Rules in Job-Shop Scheduling Problem Using Simulation: A Case Study. *International Journal of Simulation Modelling*, *11*(3), 129–140.

Kaczynski, W. H., Leemis, L. M., & Drew, J. H. (2012). Transient Queueing Analysis. *INFORMS Journal on Computing*, *24*(1), 10–28.

Karlsson, I., Papapetrou, P., & Boström, H. (2016). Generalized Random Shapelet Forests. *Data Mining and Knowledge Discovery*, *30*(5), 1053–1085.

Katoen, J. P., Brinksma, E., Latella, D., & Langarek, R. (1996). Stochastic Simulation of Event Structures. In M. Ribaudo (Ed.), *Proceedings of the 4th Workshop on Process Algebra and Performance Modelling* (pp. 21–40). CLUT.

Kayton, D., Teyner, T., Schwartz, C., & Uzsoy, R. (1996). Effects of Dispatching and Down Time on the Performance of Wafer Fabs Operating under Theory of Constraints. *Nineteenth IEEE/CPMT International Electronics Manufacturing Technology Symposium* (pp. 49–56). IEEE.

Lada, E. K., Wilson, J. R., Steiger, N. M., & Joines, J. A. (2007). Performance of a Wavelet-Based Spectral Procedure for Steady-State Simulation Analysis. *INFORMS Journal on Computing*, *19*(2), 150–160.

Laidler, G., Morgan, L. E., Nelson, B. L., & Pavlidis, N. G. (2020). Metric Learning for Simulation Analytics. In K.-H. Bae, B. Feng, S. Kim, S. Lazarova-Molnar, Z. Zheng, T. Roeder, & R. Thiesing (Eds.), *Proceedings of the 2020 Winter Simulation Conference* (pp. 349–360). IEEE.

Laidler, G. (2023). *GrahamLaidler/SimulationShapelets*. GitHub repository. https://github.com/GrahamLaidler/SimulationShapelets.

Lin, Y., Nelson, B. L., & Pei, L. (2019). Virtual Statistics in Simulation via k Nearest Neighbors. *INFORMS Journal on Computing*, *31*(3), 576–592.

Lines, J., & Bagnall, A. (2012). Alternative Quality Measures for Time Series Shapelets. *Intelligent Data Engineering and Automated Learning*, *7435*, 475–483.

Mehta, S. V., & Uzsoy, R. (1999). Predictable Scheduling of a Single Machine Subject to Breakdowns. *International Journal of Computer Integrated Manufacturing*, *12*(1), 15–38.

Morgan, L. E., & Barton, R. R. (2022). Fourier Trajectory Analysis for System Discrimination. *European Journal of Operational Research*, *296*(1), 203–217.

Mueen, A., Keogh, E., & Young, N. (2011). Logical-Shapelets: An Expressive Primitive for Time Series Classification. In C. Apte (Ed.), *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 1154–1162). Association for Computing Machinery, New York, NY.

Nelson, B. L. (2016). 'Some Tactical Problems in Digital Simulation' for the Next 10 Years. *Journal of Simulation*, *10*(1), 2–11.

Nelson, B. L., & Pei, L. (2021). *Foundations and Methods of Stochastic Simulation: A First Course* (2nd ed.). Springer Nature.

Rakthanmanon, T., & Keogh, E. (2013). Fast Shapelets: A Scalable Algorithm for Discovering Time Series Shapelets. In J. Ghosh, Z. Obradovic, J. Dy, Z.-H. Zhou, C. Kamath, & S. Parthasarathy (Eds.), *Proceedings of the 2013 SIAM International Conference on Data Mining* (pp. 668–676). SIAM.

Sargent, R. G., Goldsman, D. M., & Yaacoub, T. (2016). A Tutorial on the Operational Validation of Simulation Models. In T. M. K. Roeder, P. I. Frazier, R. Szechtman, E. Zhou, T. Huschka, & S. E. Chick (Eds.), *Proceedings of the 2016 Winter Simulation Conference* (pp. 163–177). IEEE.

Schriber, T. J., & Andrews, R. W. (1984). ARMA-based Confidence Intervals for Simulation Output Analysis. *American Journal of Mathematical and Management Sciences*, *4*(3-4), 345–373.

Shah, M., Grabocka, J., Schilling, N., Wistuba, M., & Schmidt-Thieme, L. (2016). Learning DTW-Shapelets for Time-Series Classification. In M. Marathe & M. Mohania (Eds.), *Proceedings of the 3rd IKDD Conference on Data Science, 2016* (pp. 1–8). Association for Computing Machinery, New York, NY.

Shajina, T., & Sivakumar, P. B. (2012). Human Gait Recognition and Classification Using Time Series Shapelets. In E.-S. El-Alfy & J. Aguiar (Eds.), *2012 International Conference on Advances in Computing and Communications* (pp. 31–34). IEEE.

Shedler, G. S. (1992). *Regenerative Stochastic Simulation*. Elsevier.

Ye, L., & Keogh, E. (2011). Time Series Shapelets: A Novel Technique that Allows Accurate, Interpretable and Fast Classification. *Data Mining and Knowledge Discovery*, *22*(1), 149–182.

Yuan, M., & Nelson, B. L. (1993). Multiple Comparisons with the Best for Steady-State Simulation. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, *3*(1), 66–79.

Zorko, A., Frühwirth, M., Goswami, N., Moser, M., & Levnajić, Z. (2020). Heart Rhythm Analyzed via Shapelets Distinguishes Sleep from Awake. *Frontiers in Physiology*, *10*, 1554.

Appendix A. Proof of Theorem 3.1

Theorem 3.1. *Let $y\colon [0,T] \to \mathbb{R}$ be a piecewise constant function with change times in the set $\mathcal{T} = \{0 = t_0, t_1, \ldots, t_m = T\}$ and $s\colon [0,\ell] \to \mathbb{R}$ with $\ell \leq T$ a piecewise constant function with change times in the set $\mathcal{U} = \{0 = u_0, u_1, \ldots, u_{m'} = \ell\}$. Let $U_j = \{t_i - u_j \colon i = 0, 1, \ldots, m\} \cap [0, T-\ell]$ for $j = 0, 1, \ldots, m'$, and let $\mathcal{V} = \bigcup_{j=0}^{m'} U_j$.*

Then $\|s - y([t, t+\ell])\|_1$ is linear in t for $t \in [v, v']$, where $v, v' \in \mathcal{V}$ and $(v, v') \cap \mathcal{V} = \emptyset$.

Proof. We consider the piecewise constant segments of s.

For $j = 0, 1, \ldots, m'-1$, we have $s(t) = s(u_j)$ for $u_j \leq t < u_{j+1}$. Let $\mathcal{V}_j = U_j \cup U_{j+1} \cup \{0, T-\ell\}$, and consider two points $x, x' \in [0, T-\ell]$ such that $(x, x') \cap \mathcal{V}_j = \emptyset$. We can write $x + \delta = x'$, with $\delta > 0$. For a graphical understanding, $\mathcal{V}_j$ contains the shapelet shifts such that an endpoint of the $j+1^{\text{th}}$ segment of s coincides with a change time of y.

Consider $\|s(u_j) - y([x+u_j, x+u_{j+1}])\|_1 = \int_{x+u_j}^{x+u_{j+1}} |s(u_j) - y(t)|dt$. This is the distance between the $j+1^{\text{th}}$ component of s when shifted by x and the corresponding portion of y. Since y is piecewise constant, this can be expressed as a sum of rectangular areas. The endpoints along the time axis of these rectangles are represented in the set

$$\{x + u_j\} \cup \{\mathcal{T} \cap (x + u_j, x + u_{j+1})\} \cup \{x + u_{j+1}\}. \tag{A1}$$

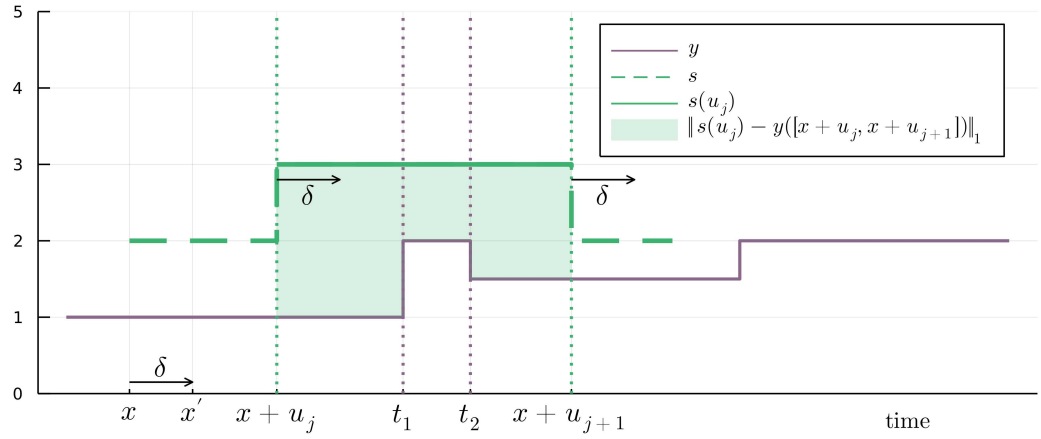

Figure A1. A labelled example illustrating the notation used throughput this proof.

Figure A1 provides a graphical illustration.

Similarly, consider $\|s(u_j) - y([x+h+u_j, x+h+u_{j+1}])\|_1$ for $0 < h \leq \delta$, with the corresponding set of endpoints

$$\{x+h+u_j\} \cup \{\mathcal{T} \cap (x+h+u_j, x+h+u_{j+1})\} \cup \{x+h+u_{j+1}\}. \tag{A2}$$

We can show that $(x+u_j, x+h+u_j) \cap \mathcal{T} = \emptyset$. This follows from a simple contradiction:

Assume that $t_i \in (x+u_j, x+h+u_j) \cap \mathcal{T}$. Then we can write $t_i = x+u_j+\epsilon$, with $0 < \epsilon < h$. Therefore $t_i - u_j = x + \epsilon \in (x, x')$. This ensures that $t_i - u_j \in [0, T-\ell]$, and since $t_i \in \mathcal{T}$, we must have $t_i - u_j \in U_j$. Therefore, we have $t_i - u_j \in \mathcal{V}_j$ and also $t_i - u_j \in (x, x')$, but this is a contradiction since $(x, x') \cap \mathcal{V}_j = \emptyset$.

Similarly, we have that $(x+u_{j+1}, x+h+u_{j+1}) \cap \mathcal{T} = \emptyset$. This follow the same logic:

Assume that $t_i \in (x+u_{j+1}, x+h+u_{j+1}) \cap \mathcal{T}$. Then we can write $t_i = x+u_{j+1}+\epsilon$, with $0 < \epsilon < h$. Therefore $t_i - u_{j+1} = x + \epsilon \in (x, x')$. This ensures that $t_i - u_{j+1} \in [0, T-\ell]$, and since $t_i \in \mathcal{T}$, we must have $t_i - u_{j+1} \in U_{j+1}$. Therefore, we have $t_i - u_{j+1} \in \mathcal{V}_j$ and also $t_i - u_{j+1} \in (x, x')$, but this is a contradiction since $(x, x') \cap \mathcal{V}_j = \emptyset$.

Therefore, we see that $\{\mathcal{T} \cap (x+u_j, x+u_{j+1})\} = \{\mathcal{T} \cap (x+h+u_j, x+h+u_{j+1})\}$. Comparing (A1) and (A2), this means that, in the graphical representation as a sum of rectangular areas, $\|s(u_j) - y([x+u_j, x+u_{j+1}])\|_1$ and $\|s(u_j) - y([x+h+u_j, x+h+u_{j+1}])\|_1$ can only differ in the widths of the first and last rectangles. Specifically, we have

$$\begin{aligned}
\|s(u_j) - y([x+h+u_j, x+h+u_{j+1}])\|_1 =& \|s(u_j) - y([x+u_j, x+u_{j+1}])\|_1 \\
&- (x+h+u_j - (x+u_j))|s(u_j) - y(x+u_j)| \\
&+ (x+h+u_{j+1} - (x+u_{j+1}))|s(u_j) - y(x+u_{j+1})| \\
=& \|s(u_j) - y([x+u_j, x+u_{j+1}])\|_1 \\
&+ h(|s(u_j) - y(x+u_{j+1})| - |s(u_j) - y(x+u_j)|)
\end{aligned}$$

Therefore, since $x < x+h \leq x'$, we see that $\|s(u_j) - y([t+u_j, t+u_{j+1}])\|_1$ is linear in t for $t \in [x, x']$.

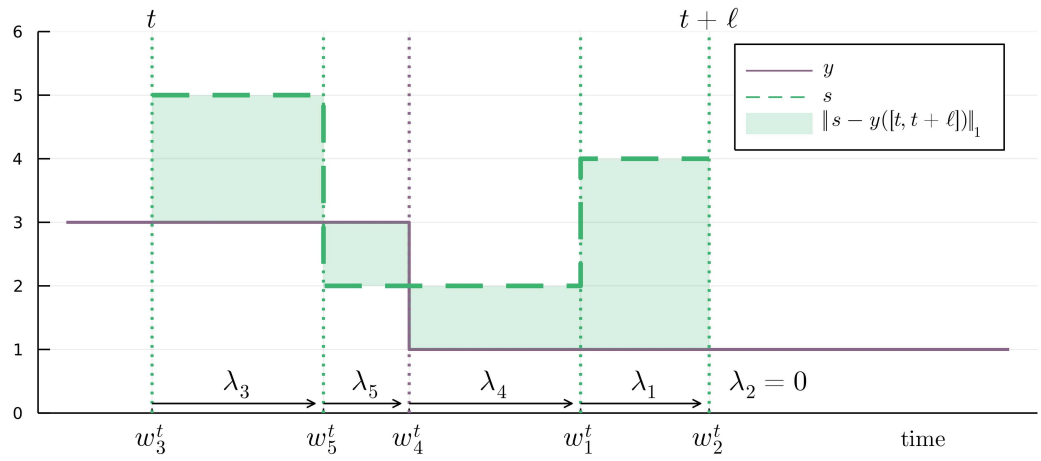

Figure B1. A labelled example illustrating the notation used throughput this proof.

Now, we can write

$$\|s - y([t, t+\ell])\|_1 = \sum_{j=0}^{m'-1} \|s(u_j) - y([t+u_j, t+u_{j+1}])\|_1.$$

Now, note that $\mathcal{V} = \bigcup_{j=0}^{m'-1} \mathcal{V}_j$, and so $\mathcal{V}_j \subseteq \mathcal{V}$ for each $j = 0, 1, \ldots, m'-1$. This means that for any pair of successive elements $v, v' \in \mathcal{V}$ with $(v, v') \cap \mathcal{V} = \emptyset$, we must also have $(v, v') \cap \mathcal{V}_j = \emptyset$ for each j. Therefore, the component $\|s(u_j) - y([t+u_j, t+u_{j+1}])\|_1$ is linear in t for $t \in [v, v']$ for each j, and we see that $\|s - y([t, t+\ell])\|_1$, as a sum of these linear components, is also linear in t for $t \in [v, v']$.

□

Appendix B. Proof of Theorem 3.3

Theorem 3.3. *Let $y\colon [0,T] \to \mathbb{R}$, $s\colon [0,\ell] \to \mathbb{R}$, and $s^c\colon [0,\ell] \to \mathbb{R}$ be as in Corollary 3.2. Let $\mathcal{W}^t = \{\mathcal{T} \cap (t, t+\ell)\} \cup \{\mathcal{U} + t\}$, with ordered elements denoted by w_i^t such that $y(w_i^t) - s(w_i^t - t) \leq y(w_{i+1}^t) - s(w_{i+1}^t - t)$ for $i = 1, 2, \ldots, m_t$. Let*

$$\lambda_i = \begin{cases} \min\{w_j^t - w_i^t \colon w_j^t \in \mathcal{W}^t, w_j^t > w_i^t\} & \text{if } w_i^t \in \mathcal{W}^t \backslash \{t+\ell\}, \\ 0 & \text{if } w_i^t = t+\ell, \end{cases}$$

and let $k^\star = \min\left\{k \in \{1, \ldots, m_t\} \colon \sum_{i=1}^k \lambda_i \geq \frac{\ell}{2}\right\}$.

Then $\min_{c \in \mathbb{R}} \|s^c - y([t, t+\ell])\|_1$ is attained by $c^t = y(w_{k^\star}^t) - s(w_{k^\star}^t - t)$.

Proof. $\mathcal{W}_t$ contains the relevant time points required to express $\|s^c - y([t, t+\ell])\|_1$ as a sum of rectangular area components. The set $\{\lambda_i\}_{i=1}^{m_t}$ contains the widths of these components. Note that the index i orders the time points in $\mathcal{W}^t$ by their series value minus their shapelet value, and the widths λ_i are ordered equivalently. Figure B1 provides an example for illustration.

We can express

$$\|s^c - y([t, t+\ell])\|_1 = \sum_{i=1}^{m_t} \lambda_i |s(w_i^t - t) + c - y(w_i^t)|.$$

We have

$$\frac{\partial \|s^c - y([t, t+\ell])\|_1}{\partial c} = \sum_{i=1}^{m_t} \lambda_i \text{sign}(c - (y(w_i^t) - s(w_i^t - t)))$$
$$\text{for } c \neq y(w_i^t) - s(w_i^t - t), i = 1, 2, \ldots, m_t.$$

Therefore, for $c \in (y(w_k^t) - s(w_k^t - t), y(w_{k+1}^t) - s(w_{k+1}^t - t))$, we can write

$$\frac{\partial \|s^c - y([t, t+\ell])\|_1}{\partial c} = \sum_{i=1}^{k} \lambda_i - \sum_{i=k+1}^{m_t} \lambda_i,$$

which implies

$$\frac{\partial \|s^c - y([t, t+\ell])\|_1}{\partial c} \begin{cases} < 0 \text{ if } \sum_{i=1}^{k} \lambda_i < \sum_{i=k+1}^{m_t} \lambda_i \\ \geq 0 \text{ if } \sum_{i=1}^{k} \lambda_i \geq \sum_{i=k+1}^{m_t} \lambda_i. \end{cases}$$

We note that since $\{\lambda_i\}_{i=1}^{m_t}$ represent the widths of a partition of $[t, t+\ell]$, we have $\sum_{i=1}^{m_t} \lambda_i = \ell$. Therefore, for $k^\star = \min\left\{k \in \{1, 2, \ldots, m_t\} \colon \sum_{i=1}^{k} \lambda_i \geq \frac{\ell}{2}\right\}$, we have

$$\frac{\partial \|s^c - y([t, t+\ell])\|_1}{\partial c} < 0 \text{ for } c \in (y(w_{k^\star-1}^t) - s(w_{k^\star-1}^t - t), y(w_{k^\star}^t) - s(w_{k^\star}^t - t)),$$
$$\frac{\partial \|s^c - y([t, t+\ell])\|_1}{\partial c} \geq 0 \text{ for } c \in (y(w_{k^\star}^t) - s(w_{k^\star}^t - t), y(w_{k^\star+1}^t) - s(w_{k^\star+1}^t - t)).$$

More generally, we will have

$$\frac{\partial \|s^c - y([t, t+\ell])\|_1}{\partial c} < 0 \text{ for } c < y(w_{k^\star}^t) - s(w_{k^\star}^t - t),$$
$$\frac{\partial \|s^c - y([t, t+\ell])\|_1}{\partial c} \geq 0 \text{ for } c > y(w_{k^\star}^t) - s(w_{k^\star}^t - t).$$

This implies that $\|s^c - y([t, t+\ell])\|_1$ is minimised at $c^t = y(w_{k^\star}^t) - s(w_{k^\star}^t - t)$.

□