Article:
Edington, L., Dervilis, N. orcid.org/0000-0002-5712-7323, Ben Abdessalem, A. et al. (1 more author) (2023) A time-evolving digital twin tool for engineering dynamics applications. Mechanical Systems and Signal Processing, 188. 109971. ISSN 0888-3270

https://doi.org/10.1016/j.ymssp.2022.109971

A time-evolving digital twin tool for engineering dynamics applications

Lara Edington[a], Nikolaos Dervilis[a], Anis Ben Abdessalem[b] and David Wagg[a,*]

[a]Department of Mechanical Engineering, University of Sheffield, Sheffield, S1 3JD, UK
[b]University of Angers, , LARIS, SFR MATHSTIC, Angers, F-49000, France

ARTICLE INFO

Keywords:
Digital twin
time-evolving
approximate Bayesian computation
optimisation
dynamics

ABSTRACT

This paper describes a time-evolving digital twin and its application to a proof-of-concept engineering dynamics example. In this work, the digital twin is constructed by combining physics-based and data-based models of the physical twin, using a weighting technique. The resulting model combination enables the temporal evolution of the digital twin to be optimised based on the data recorded from the physical twin. This is achieved by creating digital twin output functions that are optimally-weighted combinations of physics- and/or data-based model components that can be updated over time to reflect the behaviour of the physical twin as accurately as possible. The engineering dynamics example is a system consisting of two cascading tanks driven by a pump. The data received by the digital twin is segmented so that the process can be carried out over relatively short time-scales. The weightings are computed based on error and robustness criteria. It is also shown how the error and robustness weights can be used to make a combined weighting. The results show how the time-varying water level in the tanks can be captured with the digital twin output functions, and a comparison is made with three different weighting choice criteria.

1. Introduction

The digital twin concept has received significant attention from a wide range of researchers since it was first developed in 2002 in the context of product-lifecycle management [6]. The potential advantages of the digital twin have been widely recognised, and include; more affordable and efficient design, manufacturing, testing and maintenance of systems; improved useful health monitoring and life predictions; remote monitoring of systems and reduced unforeseen failures — see recent review papers [5, 7, 8, 12, 13, 14, 19, 26] and references therein. Such advantages are well-suited to a wide variety of engineering applications. In addition, the development of data-related technologies and modern computing power over recent years has further driven interest in the concept, particularly as this helps make implementations increasingly less expensive and more feasible.

Arguably the principal component in a digital twin is a computational model of the physical system (called the *physical twin*). In fact, it has been suggested that digital twins would typically be able to incorporate many distinct models that are combined in a computational representation of the system. The models involved can be physics- or data-based, or a mixture of the two, and the advantages of combining both model types are already well-recognised — see for example [26] for an (academic) digital twin example. While physics-based models are easily interpretable and able to extrapolate reliably to previously-unseen contexts (within some limitations), data-based models allow for unconstrained input-output relationships and faster, less expensive modelling, but typically cannot be used to extrapolate. Therefore, creating a digital twin that is able to combine both physics- and data-based models should enable the user to gain the benefits offered by both model types. In fact, there exist multiple discussions of digital twin interpretations which employ both model types throughout the literature — see for example [5, 8, 9, 10, 12, 13, 14, 23, 24, 26, 30], to name just a few. The idea here is to show how the models could be combined as time evolves.

There is already a large field of work on combining models, generally called *ensemble modelling*, and including techniques such as model averaging, boosting, bagging, stacking etc. — see overviews by [2, 32], and references

*Corresponding author
ljedington1@sheffield.ac.uk (L. Edington); N.Dervilis@sheffield.ac.uk (N. Dervilis); anis.ben-abdessalem@univ-angers.fr (A.B. Abdessalem); David.Wagg@sheffield.ac.uk (D. Wagg)
ORCID(s): 0000-0002-5712-7323 (N. Dervilis); 0000-0002-7266-2105 (D. Wagg)

therein. However, there are some important differences between what is shown here for a digital twin (as we define it) and ensemble methods. Generally, ensemble methods are based on having access to the complete data set. In other words the techniques are applied in a post-processing setting. This approach can be applied in a digital twin. However, if data are being gathered continuously, there will need to be a clearly-defined rationale regarding when a data set is sufficiently 'complete' so that the post-processing can be started (which will be specific to the context at hand). In addition to post-processing methods, there are some methods that can be applied in real-time (or near real-time), and some, such as those built on Kalman filters, can be used to deal with types of ensemble modelling in a near real-time context — see for example [31]. However, filter techniques are usually considering based on sequential time steps — although there is a wide literature of extension to these methods [4, 17] — particularly related to data assimilation and forecasting, which we see as a related, but distinctly different set of methods from those being considered here.

The approach taken here is designed to operate on data segments as they are recorded (or received) by the digital twin. The intention is to develop a method that fits between the (near) real-time (e.g. online) and post-processing time-frames. The potential benefit is that it offers the user a relatively rapid insight into the behaviour of the system over the short term, and which model combinations might be used with most confidence (depending on the specific context at hand). To achieve this aim, a method for combining models based on model weightings is developed. This approach has similarities with some existing ensemble methods, such as the mixture of experts approach [32]. However, as the digital twin is expected to operate in a time-evolving context, existing methods for choosing weights for combining models cannot be easily applied. Instead, a weighting selection method is developed based on relating the underlying dynamic and statistical models. The Physinet digital twin proposed by Sun & Shi [23] proposes an approach of this type which combines two models, one physics-based and a neural network model, as a weighted combination in order to make predictions of the physical twin.

In this paper we incorporate an approach which includes both aleatory and epistemic uncertainties. As a result, our approach also has similarities to the methods used for model calibration and other uncertainty quantification procedures — see for example [11, 21] and references therein. Application of the model calibration method to digital twin applications (in related but different contexts to the current work) has been previously discussed by several authors [25, 26, 27]. In particular, Ward et al. [27] developed a continuous-calibration approach for a digital twin, and compared the use of a particle filtering methodology and a sequential implementation of the Bayesian calibration approach introduced by Kennedy and O'Hagan [11], in calibrating a physics-based model with dynamic parameters.

Another important characteristic of a digital twin is a capability to model complex engineering (or other) systems. In this paper, relatively simple, proof-of-concept example will be shown, but in a practical digital twin implementation it would be quite typical that the physical twin would have some more significant complexity (although this is always somewhat context dependent). For this reason, approximate Bayesian computation (ABC) is used for the physics-based model in this work, on the premise that closed-form expressions are only rarely available in the context of digital twins. As already mentioned, the context-specific nature of a digital twin makes each one highly bespoke, and it is very likely that other choices of models, or weighting choice criteria will be more appropriate in those cases. As a result, the overall approach developed here should translate to a new context, assuming that the underlying ethos is maintained.

The details of the development of the digital twin output functions, beginning with the theoretical background, is presented in Section 2. In Section 3 the digital twin is applied to an engineering example, which is two cascading water tanks with water levels driven by a pump, where certain inputs produce overflow effects. Finally, the overall performance of the presented new methods are discussed in Section 4, and Section 5 summarises the conclusions and future work.

2. A theoretical model of a time-evolving digital twin

2.1. Digital twin output functions

To start with a finite set of $N_{\mathbf{z}}$ quantities of interest (QoI) will be defined, that can be observed from the physical twin, and these will form a vector $\mathbf{z} \in \mathbb{R}^{N_{\mathbf{z}} \times 1}$. In order to observe the time evolution of the digital twin, the n^{th} QoI, $z_i^{(n)} \in \mathbf{z}$, (for $n = 1, 2, 3...N_{\mathbf{z}}$) element of $\mathbf{z}$ is measured (e.g. sampled) at time step i from the physical twin, and there is a corresponding discrete time series, $t_i \in [t_{start}, t_{end}]$, with a fixed time-step of Δt. The digital twin will use a specified combination of physics-based, data-based, and/or hybrid models to compute an approximation to the state of the physical twin QoIs at a particular time instant, t_i, and the output of of the digital twin will be given by

$$\mathbf{y} = \boldsymbol{\eta}(\mathcal{M}, \mathcal{D}, \mathcal{T}, \boldsymbol{\chi}, t_i) \tag{1}$$

where $y_i^{(n)} \in \mathbf{y}$ is the n^{th} scalar output, and $\eta_i^{(n)} \in \boldsymbol{\eta}$ is the corresponding *digital twin output function*. Each digital twin output function is assumed to be a function of one or more of the N_p models $\boldsymbol{M}_p \in \mathcal{M}$, where $\mathcal{M}$ is the model library containing all physics-based, data-based and hybrid (e.g. grey-box) models. Data-based and grey-box models are dependent on data sets that are contained in the library of data sets for the digital twin denoted $\mathcal{D}$. The time-based parameters $\{t_{start}, t_{end}, \Delta t\}$ are contained in the time-base library, $\mathcal{T}$, and any hyper-parameters for the digital twin (defined below) are contained in the vector $\boldsymbol{\chi}$.

The outputs of each of the *p dynamic models*, $\boldsymbol{M}_p$, are represented (dropping the *n* superscripts) by

$$\hat{y}_{i,p} = \hat{\eta}_{i,p}(\mathbf{x}_i, t_i; \boldsymbol{\theta}_p, \mathbf{u}_p, D_p, \boldsymbol{\psi}_p) \qquad \text{for } p = 1, 2, 3...N_p \tag{2}$$

where $\hat{y}_{i,p}$ is the p^{th} scalar model output at time t_i, and $\hat{\eta}_{i,p}$ is the corresponding *model output function*. Each model output is a function of the state vector $\mathbf{x}$, the physical parameters vector $\boldsymbol{\theta}_p$, the input/control signals vector $\boldsymbol{u}_p$ and time, t_i. In addition, any data sets required by the model are contained in D_p, and if model hyperparameters are needed, they are contained in the vector $\boldsymbol{\psi}_p$. Although parameters and inputs can vary between models, in this work all models are assumed to have the same state vector, $\mathbf{x}$. Note that the index notation here relates to the i^{th}observation of physical twin process z, and so $\mathbf{x}_i$ defines the i^{th} iteration of the state vector $\{\mathbf{x} : x_k \in \mathbf{x}\}_i$ for $k = 1, 2, 3....N_k$ where N_k is the total number of states.

The QoI's from the physical twin and the outputs of the digital twin are related via a *statistical model* [11, 21]. For the i^{th}observation of the n^{th} QoI, z_i^n, of the physical twin, the statistical model for the digital twin (omitting the *n* superscripts),will be defined as

$$z_i = \zeta_i + e_i = \eta_i(\mathcal{M}, \mathcal{D}, \mathcal{T}, \boldsymbol{\chi}, t_i) + \delta_i + e_i, \tag{3}$$

where the (unobservable) true process of the physical twin is ζ_i, and e_i is the observation (e.g. measurement) error. The digital twin output function η_i at the i^{th}observation (computed using (1) and (2) as will be explained in the next Section) is the best estimate of the QoI at time t_i. The inadequacy (or deficiency) of η_i is represented by δ_i. In some interpretations δ_i and e_i are combined into a single term — see for example [9] — and it is assumed here that $\delta_i + e_i$ captures the combined errors and uncertainties relating to the corresponding z_i.

Equation (3) has been used extensively for Bayesian calibration applications following the work of [11] (see also [27] and the references therein). In that context, [27], have adapted the Bayesian calibration idea to a digital twin application, where the calibration occurs sequentially. Here, the statistical model (3) will be used in the context of a time-evolving digital twin to compare the QoIs with the digital twin output functions whilst also accounting for the most important errors and uncertainties present in the problem. Specifically these are:

A1. observation (e.g. measurement) error is represented by e_i
A2. model form errors are assumed to be captured by δ_i
A3. parameter uncertainties can be included in the physical parameter vectors $\boldsymbol{\theta}_p$
A4. numerical errors (sometimes treated as a separate term e.g. [27]) are assumed to be captured by δ_i

In this work we will use the magnitude of $\delta_i + e_i$ to define the performance of each model, and then select appropriate weightings (described in Section 2.2). Estimating the parameter uncertainties can be carried out using a range of techniques, and the exact choices are typically specific to the context of the application being considered. For this reason, we will delay the description of the methods used for modelling and estimating parameter uncertainties until Section 3.

2.2. Performance measures and weighting functions

First note that equation (3) can be rearranged such that for the i^{th}observation of the n^{th} QoI, z_i^n,

$$\delta_i + e_i = z_i - \eta_i \tag{4}$$

where superscript *n* has been dropped. The intention here is to compare between the QoIs of the physical twin, and the output functions of the digital twin. As the digital twin is time evolving, one approach is to make a comparison for each time-step (e.g. tracking control, in a control engineering context). Adopting the definition proposed by [30], a digital twin output function would then be an ϵ-mirror (meaning an appropriate representation) of the physical twin, as long as $|\delta_i + e_i| < \epsilon$, (via equation 4) where ϵ is defined depending on the specific context of the application — see [30], for more details.

A second approach (also discussed by [30]) is to compare the behaviour over a larger number of time-steps. To do this a set of N data points (which we call a data segment, $\boldsymbol{d}_s$), will be used to relate the recorded QoI of the physical twin (e.g. the true process) to a digital twin output function, and therefore quantitatively compare the physical and digital twins. As with the approach described by [27] we assume that that parameters remain constant over the data segments, but may change over longer timescales.

In order to create a performance measure, equation (4) can be used to compute the root-mean-square error (RMSE) for all observations in a data segment $\boldsymbol{d}_s$ made up of N data points, for the n^{th} QoI using

$$RMSE_{s,n} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\delta_i + e_i)^2} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(z_i - \eta_i)^2} = \sqrt{C_{s,n}} \tag{5}$$

which can be computed using the observations z_i and the corresponding digital twin output function η_i (because δ_i and e_i are not known), and where $C_{s,n}$ is the total mean-square error for the digital twin output function in data segment $\boldsymbol{d}_s$ for the n^{th} QoI. Note that although RMSE can be used for the purposes of performance measures and choice of weighting functions, it is more common in many applications (see for example [29]) to use a normalised version of the mean-square error (e.g. $C_{s,n}$) given by

$$NMSE_{s,n} = \frac{100}{N\sigma_z}\sum_{i=1}^{N}(z_i - \eta_i)^2 = \hat{C}_{s,n} \tag{6}$$

where σ_z is the variance of the nth measured QoI.

The digital twin output functions defined in equation (1) can be comprised of more than one computational model (and here it is assumed there is a combination of physics- and/or data-based model components with output functions defined in equation (2)). To combine these multiple components into one digital twin output function, a series of $\rho_{s,p} \in \chi$ weighting functions are introduced for each of the N_p model output functions and the s^{th} data segment $\boldsymbol{d}_s$. Equation (3) can therefore be modified to show the combination of P physics- and D data-based models, for the $\rho_{s,p}$ weightings (one for each of the $P + D = N_p$ models, and with $p = 1, 2, 3..., N_p$) such that the combined model outputs then produce one digital twin output function for the n^{th} QoI, as given in equation (1). This weighted combination of model output functions has the form (omitting the n superscripts) of

$$\begin{aligned} z_i &= \rho_{s,1}\hat{\eta}_{i,1} + \rho_{s,2}\hat{\eta}_{i,2} + ... + \rho_{s,p}\hat{\eta}_{i,p} + ... + \rho_{s,N_p}\hat{\eta}_{i,N_p} + \delta_i + e_i \\ &= \eta_i(\mathcal{M}, \mathcal{D}, \mathcal{T}, \chi, t_i) + \delta_i + e_i \end{aligned} \tag{7}$$

where $\hat{\eta}_{i,p}$, are the model output functions given in (2) that are each predicting the *same*, e.g. the nth, output. The additive relationship $\eta_i = \rho_{s,1}\hat{\eta}_{i,1} + .. + \rho_{s,N_p}\hat{\eta}_{i,N_p}$ relates the model output functions to the digital twin output function, and is assumed to hold for each of the n QoIs in $\boldsymbol{N_z}$. Note that in this work, it is assumed that the computation time for each model is sufficiently fast so that the model outputs are available when required.

Notice that the formulation given in (7) allows several possibilities depending on how the weighting functions are chosen. For example, models from the model library can be selected (or deselected) using the weights. If just a single model is selected, then the model calibration methods described by [11, 27] could potentially be applied. If multiple weightings are used (and a post-processing setting is available), then a range of ensemble types methods may also become applicable, depending on the precise context being used [32].

In this work, we assume that the p^{th} model component at time step t_i will have states $\mathbf{x}_i$, and a weight $\rho_{s,p} \in [0, 1]$ such that $\sum_{p=1}^{N_p}\rho_{s,p} = 1$. Once the component weights are chosen for a data segment, $\boldsymbol{d}_s$, it is possible to define a *physics-to-data model fraction* of the n^{th} digital twin output function, and s^{th} data segment as,

$$\Upsilon_{s,n} = \sum_{j=1}^{P}\rho_{s,j}^{(P)} \qquad \text{for} \quad j = 1, 2, ...P \tag{8}$$

where $\rho_{s,j}^{(P)}$ are the weightings for just the physics models. Here $\Upsilon_{s,n} \in \chi$ is defined as the proportion of models that are physics-based in the digital twin output function (other definitions could be used depending on the circumstances,

e.g. the proportion of data-based models is $1 - \Upsilon_{s,n}$ in this case, as we consider only physics- and data-based models). Hence $\Upsilon_{s,n}$ provides a measure of the composition of the resulting digital twin, from which we can infer several properties. For example:

B1. The relative trust ascribed to each of the two model types (physics- and data-based) at the i^{th} observation, or in the s^{th} data segment — which can also be considered as a measure of model error and/or robustness to uncertainties, as will be discussed below

B2. Whether the digital twin could be used to extrapolate reliably to previously-unseen contexts (within some limitations) — which would require a (relatively) high $\Upsilon_{s,n}$ value, so that the digital twin is primarily physics-based

B3. Information about the relative levels of deficiency of different models — a factor that will typically be accounted for when choosing the $\rho_{s,p}$ values, but not always quantitatively

2.3. Choosing the weighting functions

2.3.1. Error-based weightings

The weights will be chosen in order to deliver some of the three desirable properties, B1, B2 & B3 listed above. Here we will focus on property B1, and firstly on choosing optimal weights to reduce the error between the n^{th} QoI, z_i and associated output function, η_i. Due to the relationship in equation (4) this can also be considered to be the weights that minimise the combined uncertainty $\delta_i + e_i$ for each of the digital twin output functions.

To compute the error for each data segment of N points the p^{th} model output function will be compared directly with the corresponding QoI, z_i in the form of a root-mean-square discrepancy function equal to,

$$RMSE_{s,p} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(z_{i,p} - \hat{\eta}_{i,p})^2} = \sqrt{C_{s,p}} \tag{9}$$

where $C_{s,p}$ is the mean-square error of the output of model $\boldsymbol{M}_p$ for the data segment $\boldsymbol{d}_s$. Based on the output of equation (9) there are multiple ways to choose weightings (for example by using some of the ensemble type methods described in [32]). Here, we will adopt two approaches, to give an indication of some possible methods (the exact choice of approach will be determined by the context of the application).

In order to demonstrate the concept, as applied to a time-evolving digital twin, the first approach taken here will be to use a weighted average method — see for example [15, 32] — where each of the z_i QoI's is assumed to be a random variable. Therefore, in terms of obtaining the weight values we wish to minimise the uncertainties in the weighted sum of the model errors computed in equation (9)

$$\sum_{p=1}^{N_p} \rho_{s,p}\sqrt{C_{s,p}} \quad \text{subject to the constraints that} \quad \sum_{p=1}^{N_p} \rho_{s,p} = 1 \quad \text{and} \quad \rho_{s,p} \in [0, 1] \tag{10}$$

In order to demonstrate the concept whilst also keeping the details relatively straightforward, in the examples presented in the later part of this paper we will restrict ourselves to the combination of just two models. One will be physics-based and the other data-based, such that $P = 1$, $D = 1$ and $N_p = 2$. In this case, a simple way to compute the weighting functions is to equalise the errors of the two models over the data segment [15]. This gives weighting values of

$$\beta_1 = \frac{\sqrt{C_{s,2}}}{\sqrt{C_{s,1}} + \sqrt{C_{s,2}}}, \quad \beta_2 = \frac{\sqrt{C_{s,1}}}{\sqrt{C_{s,1}} + \sqrt{C_{s,2}}} \tag{11}$$

so that we choose the weights based on

$$\rho_{s,p} = \begin{cases} \beta_p, & \text{if } C_{s,\beta} < C_{min} \\ 0, & \text{if } C_{s,\beta} > C_{min} \text{ and } C_{s,p} \neq C_{min} \\ 1, & \text{if } C_{s,\beta} > C_{min} \text{ and } C_{s,p} = C_{min} \end{cases} \qquad \text{for } p = 1, 2 \quad \text{and} \quad C_{min} = \min[C_{s,1}, C_{s,2}] \tag{12}$$

Here $C_{s,\beta} = \sum_{p=1}^{N_p} \beta_p \sqrt{C_{s,p}}$ (from equation (10)), and C_{min} is the mean-square error of the single-most best-fitting model, $\boldsymbol{M}_p$, for data segment $\boldsymbol{d}_s$. With this weighting selection method, the optimal digital twin output is automatically

chosen from either the combination of the two models or the better performing model, should one of them have a smaller mean-square error than the combined digital twin output function itself. An example will be shown in Section 3.3.

2.3.2. Robustness-based weightings

The second criterion mentioned in B1 above is maximising robustness of the digital twin's outputs to uncertainty. Specifically, we are interested in uncertainties in the physical parameters in $\boldsymbol{\theta}_p$, for the p^{th} model, as mentioned in point A3 above. In order to consider this type of scenario, further assumptions are needed about the p^{th} model, $\boldsymbol{M}_p$. Namely, that the parameters are modelled using some form of uncertainty analysis, for example as probability distributions or ranges of possible values. The model would also typically be expected to produce an output that can be treated as a random variable. So, keeping the assumption made above, that $z_i^{(n)}$ is a random variable, we will assume that the corresponding model output, $\hat{y}_{i,p}^{(n)}$ from equation (2) is also a random variable. Furthermore, for this work we assume that all $\boldsymbol{M}_p$ models have these properties.

There are many approaches to estimating the level of uncertainty in a specific model. Here (as a demonstration of the concept), a minimax decision-making approach is used to carryout this task for each of the $\boldsymbol{M}_p$ models. Minimax is a non-probabilistic method of choosing one of multiple possible decisions, with the option selected being that which minimises the maximum cost. In this work the models' physical parameters, contained in $\boldsymbol{\theta}_p$, are assumed to be the only source of uncertainty. As minimax is a non-probabilistic theory, only ranges of possible parameter values (and no probability distributions) are required. Within these ranges of possible parameter values for each model, an optimisation process determines the largest cost of that model's response. Here, the optimisation is carried out using a self-adaptive differential evolution (SADE) genetic algorithm [22]. Having computed the models' worst-case responses, the component weights are then chosen to minimise the resulting digital twin output function's NMSE. The weight selected with this robustness criterion for model p is denoted $\rho_{s,p}^r$, where $\rho_{s,p}^r \in [0, 1]$ such that $\sum_{p=1}^{N_p} \rho_{s,p}^r = 1$. An example will be discussed in Section 3.4.

2.3.3. Combined error and robustness-based weightings

Computing weighting values based on different criteria leads to the possibility of combining the weights themselves into an ensemble parameter. For example, to compute one combined weight, $\rho_{s,p}^c$, from the error- and robustness-optimised weights, $\rho_{s,p}^e$ and $\rho_{s,p}^r$, for each component p. This can simply done by taking a linear combination:

$$\rho_{s,p}^c = \alpha\rho_{s,p}^e + (1-\alpha)\rho_{s,p}^r \tag{13}$$

An appropriate method can then be selected by the user in order to choose the value of α depending on how they need to prioritise each of the optimisation criteria. This may vary with the application and the level of risk deemed acceptable, and prioritising robustness to uncertainty would potentially lower the accuracy of the digital twin's output. An example will be presented in Section 3.4.5.

Note that (depending on the exact context – discussed further in Section 3), all these weighting choices will typically tend to favour data-based models, as they are built based on matching an input-output relationship. Choosing weights based on properties B2, and B3 could potential change this effect, but will not be considered in detail here.

Having defined the performance measures and weighting functions, the temporal evolution of the digital twin can be finalised. The digital twin should be able to continuously (typically periodically) update over time, in order to reflect the time dependent changes occurring in the physical twin. In line with this aim, some or all of the physics-based models may be sequentially updated to incorporate any new physical knowledge of the twinned system. Any data-based models must also be re-trained, and the optimal model weights must be iteratively recalculated according to data. The user would decide on a suitable iteration frequency for the application of the digital twin; higher-frequency iterations would produce a faster-updated digital twin, but would be more computationally intensive. Next we consider an example.

3. Example application: A cascaded tanks system

In this Section, a physical twin system that includes both nonlinearity and uncertainty is used to demonstrate the how the concepts described above can be applied to an engineering application. Specifically the cascading tanks

system detailed in [18] is selected as an example for a time-evolving digital twin system. The cascading tank system is comprised of two water tanks, a reservoir and a pump. A diagram of the system is shown in Fig 1. An input controls

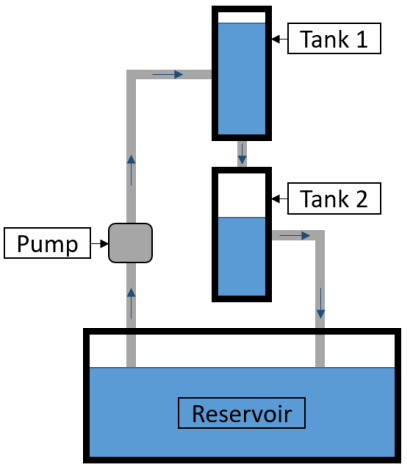

Figure 1: Schematic diagram of the cascading tanks system, as described by [18].

the pumping of water from the reservoir into Tank 1, which then flows into Tank 2 below it, and finally back into the reservoir. It is possible for the tanks to overflow under larger inputs, and Tank 1 may overflow into Tank 2, as well as into the reservoir, which makes modelling the physics of the system particularly difficult. Fig 2 displays a plot of a recorded input signal, u_1, applied to the system. This input is assumed to be an arbitrary input that is not influenced by the digital twin. In other scenarios, the input could be under the control of the digital twin. It can be noted from Fig

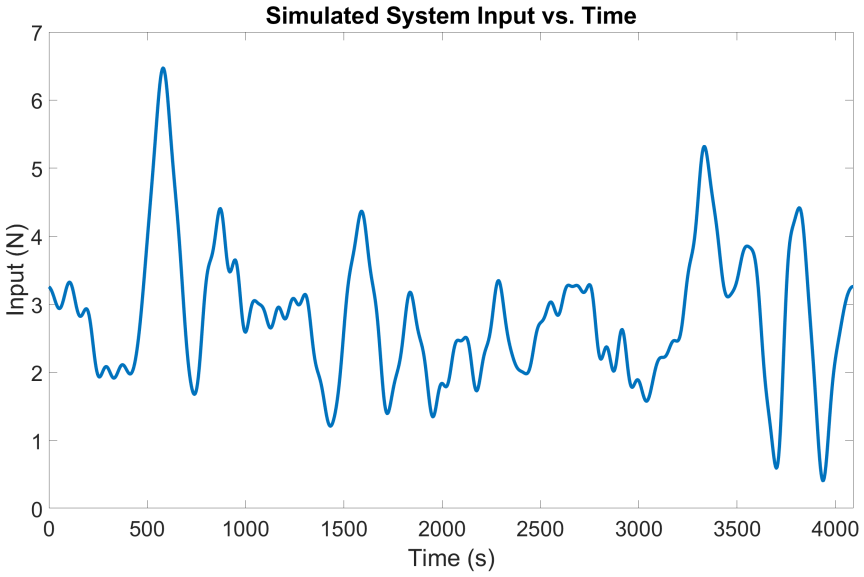

Figure 2: Recorded input, u_1, applied to cascading tanks system. Data available from [18].

2 that $t_{start} = 0$, $t_{end} = 4096$s and $\Delta t = 4$s for this example.

3.1. Physics-based models

As a starting point we define two QoI's which are the water levels of Tank 1 & 2. Therefore $z_i^{(1)}$ = water level Tank 1 and $z_i^{(2)}$ = water level Tank 2, such that $\mathbf{z} = [z_i^{(1)}, z_i^{(2)}]^T$ and $N_{\mathbf{z}} = 2$. The system can be approximately modelled, by neglecting the overflow effects, using a reduced order nonlinear set if ordinary differential equations that will be defined as model M_1 in the model library $\mathcal{M}$. The model is defined as [18]

$$M_1 = \begin{cases} \dot{x}_1(t) &= -k_1\sqrt{x_1(t)} + k_4 u(t) + w_1(t) \\ \dot{x}_2(t) &= k_2\sqrt{x_1(t)} - k_3\sqrt{x_2(t)} + w_2(t) \end{cases} \quad \text{and} \quad y_2 = \hat{\eta}_{i,1}^{(2)} = x_2(t) \tag{14}$$

where $x_1(t)$ and $x_2(t)$ are the water levels of the upper and lower tanks respectively, $u(t) = u_1$ is the system input, $\boldsymbol{\theta}_1 = [k_1\ k_2\ k_3\ k_4]^T$ is the vector of the parameters and $w_1(t)$ and $w_2(t)$ are noise terms (and t is continuous time).

However when $x_2 > 10$ Tank 1 will overflow into the reservoir. As a result, [28] developed a model that incorporates a fifth parameter to represent overflow effects which can be denoted as

$$M_2 = \begin{cases} \dot{x}_1(t) &= -k_1\sqrt{x_1(t)} + k_4 u(t) + w_1(t) \\ \dot{x}_2(t) &= \begin{cases} k_2\sqrt{x_1(t)} - k_3\sqrt{x_2(t)} + w_2(t) & x_1(t) \le 10 \\ k_2\sqrt{x_1(t)} - k_3\sqrt{x_2(t)} + k_5 x(t) + w_3(t) & x_1(t) > 10 \end{cases} \end{cases} \quad \text{and} \quad y_2 = \hat{\eta}_{i,2}^{(2)} = x_2(t) \tag{15}$$

where k_5 is the additional parameter and $w_3(t)$ an additional noise term, such that $\boldsymbol{\theta}_2 = [k_1\ k_2\ k_3\ k_4\ k_5]^T$ for this model. When the system exhibited overflow, this fifth term could take a significant value — otherwise, it could equal zero so that the model would be equivalent to that described by equations (14).

Although M_1 and M_2 are relatively simple models, they incorporate both nonlinear effects, and terms to represent the uncertainties inherent in the physics. As a result, an approximate Bayesian computation (ABC) algorithm is used as a way to produce estimates for the physical parameters, $\boldsymbol{\theta}_1$, in M_1 (or $\boldsymbol{\theta}_2$, in M_2). Further details will be given in the relevant subsections below.

3.2. Data-based model

For the data-based model (DBM), a nonlinear auto-regressive exogeneous (NARX) neural network (NN) model, labelled M_3, was used to model Tank 2's water level as a function of lagged versions of the system input and response. The M_3 model is represented by

$$M_3 = \{\ x_2(t_i) = F(x_2(t_i-1), x_2(t_i-2), ..., x_2(t_i-n_x); u_1(t_i), u_1(t_i-1), ..., u_1(t_i-n_u)) \quad \text{and} \quad y_2 = \hat{\eta}_{i,3}^{(2)} = x_2(t) \tag{16}$$

where $x_2(t_i)$ and $u_1(t_i)$ are the level of Tank 2 and input respectively at time step t_i, n_x is the maximum output time lag and n_u the maximum input time lag. The function F is nonlinear, and the NARX-NN is modelled by a neural network. Further details on NARX models may be found in [3]. In this work, a 3-layer NN with 5 hidden nodes was used for each segment's NARX model. Within each segment, n_x and n_u were set to be equal – the number of maximum lags was determined by initialising it to 1 and increasing by 1 as long as the error of the NARX-NN's output was larger than $NMSE = 5$.

A NARX model can make two types of prediction: one step ahead (OSA) — predictions are based on previous measured data points — and model predicted output (MPO), where predictions are based on previous predicted data points. In this work, the NARX-NN model uses MPO outputs to allow the digital twin to predict (theoretically) indefinitely into the future instead of only one point ahead. It should be noted that unlike OSA, any error in MPO predictions is propagated through all future time steps and so MPO is considered a stronger test of the NARX model. For the digital twin, MPO predictions were used as the data-based model component's response, as a significant advantage of the digital twin technology is to forecast the physical twin's future behaviour, which would need to be several time steps ahead.

The neural network architecture chosen for M_3 in all examples presented in this paper was a three-layer net with ten hidden nodes. This architecture was chosen as a trade-off between the simplicity required to limit computational expense and the complexity required to effectively model the systems. It would be costly to trial various networks for every iteration of the digital twin framework, and choosing one architecture to generalise to all iterations was deemed justifiable for the current purpose, and produced satisfactory results. However, it is possible to treat the number of hidden nodes as a model hyper-parameter to be tuned, at a computational expense, where necessary.

For simplicity, the number of maximum lags of both input and output for the NARX model were assumed to be equal. The maximum lag was initially set to 1 and the resulting neural network was trained. If the normalised mean-square error (NMSE) — the mean-square error divided by the variance of the training data — of the trained model's validation MPO response was greater than an acceptable threshold (equal to 1), the number of lags was increased by 1 and the model was retrained. This was repeated until the NMSE was smaller than the specified threshold to determine the number of lags.

The data segments, $\boldsymbol{d}_s$ were defined by splitting the tanks input, u_1 and Tank 2 water level (e.g. QoI) time series, $z_i^{(2)}$ consisting of 1024 data points into eight equal-sized segments of 128 data points (that is, 64 alternating training and validation points). Therefore the data library for this digital twin was $\{\boldsymbol{d}_1, \boldsymbol{d}_2, ..., \boldsymbol{d}_8\} \in \mathcal{D}$ and $\{\boldsymbol{u}_1, \boldsymbol{u}_2, ..., \boldsymbol{u}_8\} \in \mathbf{u}$ where each $\boldsymbol{d}_s$ and $\boldsymbol{u}_s$ is a 128×1 array of data.

3.3. Error-based weightings 1 — weighted averages

In this scenario, the physics-based model is chosen to be M_2 and the data-based model is M_3, so the statistical model for $n = 2$ is given by

$$z_i^{(2)} = \rho_{s,2}\hat{\eta}_{i,2}^{(2)} + \rho_{s,3}\hat{\eta}_{i,3}^{(2)} + \delta_i + e_i = \eta_i^{(2)} + \delta_i + e_i \tag{17}$$

where δ_i represents the unknown combined model inadequacies, and $\eta_i^{(2)} = \rho_{s,2}\hat{\eta}_{i,2}^{(2)} + \rho_{s,3}\hat{\eta}_{i,3}^{(2)}$ is the combined digital twin output function. Notice that the subscript, p for ρ_p and $\hat{\eta}_{i,p}$ relates to which model in the model library is being used. For this example there are three models in the library, $M_1, M_2, M_3 \in \mathcal{M}$. So another way to interpret equation (17) is that the weightings are being used to select two out of the three available models, and therefore in this case $\rho_{s,1} = 0$. Note also that $P = 1$, $D = 1$, and so $N_p = 2$ in this example.

The physics-based model, M_2, contains uncertainties, and therefore the parameters were estimated using ABC. A difficulty in estimating the model parameters is that only the second tank's water level was measured and could be used to ascertain a sampled model's dynamic behaviour. To deal with this, M_2 was simulated with the physical parameters, $\boldsymbol{\theta}_2$, sampled from an estimated range. Each parameter sample was used to estimate the water levels, comparing the simulated and measured second tank level in determining the model mean-square error.

There were two data sets provided with the benchmark system, recorded over different time periods. It was assumed that one data set was available to the user before applying the digital twin (training) and the second was recorded as the digital twin was applied (testing). It is the second data set that the digital twin would aim to mirror. This choice regarding the separation of the data sets into training and testing is essentially an arbitary choice based on the requirement to have a distinction for the methods being used here. Other choices may be more appropriate for other cases. Recall from Section 2.3.1 that each model will be compared with the data recorded from the physical twin (the training and testing data in this case). Therefore, an initial ABC simulation was implemented on the first data set, using normal prior distributions, in order to inform the prior distributions used for the digital twin ABC simulations on the second data set.

To infer the model parameters, an ABC algorithm with an accept-reject mechanism is employed. Given the observations $\boldsymbol{d}_s$, from the sth data segment, the basic principle consists in iterating the following steps until one gets N_o particles:

- **Step. 1** Generate a candidate value for the parameter vector $\boldsymbol{\theta}^* \sim \boldsymbol{\pi}(\boldsymbol{\theta})$ from the prior.
- **Step. 2** Generate a data set $\boldsymbol{y}_s^* \sim M_p(\boldsymbol{\theta}^*)$ using the output from the pth model.
- **Step. 3** Accept $\boldsymbol{\theta}^*$ if $\kappa(\boldsymbol{d}_s, \boldsymbol{y}_s) \leq \varepsilon_t$.
- **Step. 4** If rejected, go to **Step. 1**.

where $\kappa(\cdot)$ is a distance function, measuring closeness of simulated and observed data, and ε_t is a threshold parameter.

The prior distributions that were chosen for the five parameters of the physics-based model component were:

$$\boldsymbol{\pi}(\theta_2^{(1)}) \sim \mathcal{N}[0.05, 0.01]$$
$$\boldsymbol{\pi}(\theta_2^{(2)}) \sim \mathcal{N}[0.055, 0.01]$$
$$\boldsymbol{\pi}(\theta_2^{(3)}) \sim \mathcal{N}[0.05, 0.01]$$

Table 1
Parameter values of the physics-based model component over all segments.

Segment	$s=1$	$s=2$	$s=3$	$s=4$	$s=5$	$s=6$	$s=7$	$s=8$
$\theta_2^{(1)}$	0.0588	0.0600	0.0548	0.0557	0.0564	0.0549	0.0565	0.0559
$\theta_2^{(2)}$	0.0543	0.0537	0.0514	0.0495	0.0497	0.0500	0.0515	0.0510
$\theta_2^{(3)}$	0.0516	0.0585	0.0547	0.0555	0.0561	0.0555	0.0550	0.0547
$\theta_2^{(4)}$	0.0503	0.0503	0.0504	0.0500	0.0494	0.0494	0.0499	0.0509
$\theta_2^{(5)}$	0.0001	0.0017	-0.0002	-0.0001	-0.0004	-0.0002	0.0014	0.0005

Table 2
Evolution of the overflow parameter, $\theta_2^{(5)}$, in the physics-based model component over all segments.

Segment, s	1	2	3	4	5	6	7	8
$\theta_P^{(5)}$	-4.1×10^{-5}	0.0017	-0.0004	-0.0002	0.0003	-0.0003	0.0015	0.0007

Table 3
Evolution of original and updated physics-to-data model fractions in the digital twin over time.

Segment, s	1	2	3	4	5	6	7	8
$\Upsilon_{s,n}$ with weights from equation (11)	0.1046	0.0870	0.2425	0.1579	0.0572	0.1561	0.1862	0.2150
$\Upsilon_{s,n}$ with weights using equation (12)	0.0983	0.0	0.0	0.0	0.0	0.0	0.0	0.0

$$\boldsymbol{\pi}(\theta_2^{(4)}) \sim \mathcal{N}[0.055, 0.01]$$
$$\boldsymbol{\pi}(\theta_2^{(5)}) \sim \mathcal{N}[0, 0.005]$$

which were chosen to reflect relatively-weak prior knowledge. The distance function, $\kappa(\cdot)$, used was the mean-square error of the model's estimated second tank water level compared to the measured water level; the tolerance used was $\varepsilon_t = 1$ and 1000 particles were accepted. Normal posterior distributions were fitted to all parameters of the accepted particles.

For the data-based model, $\boldsymbol{M}_3$, a three-layer NARX neural network with 10 hidden units, was formed using lagged versions of the second tank's water level and $u(t)$ as inputs to the neural network, and the second tank's level as the output (e.g. equation 16). The training data were normalised to span the range of [0,1] to avoid saturation of the network. Again, the number of maximum lags of the input and output were assumed to be equal for simplicity, and were initially set to 1. This lag number was increased by 1 if the normalised mean-square error of the MPO of the resulting trained network was larger than an acceptable threshold, and the new network was retrained. For the hidden nodes a greedy trial-and-error search was used to find the optimal output. The NARX models' MPO predictions were used as the responses of the data-based model component.

Then $\boldsymbol{M}_2$ and $\boldsymbol{M}_3$ were re-trained over each of the 8 data segments $\boldsymbol{d}_s$, and their responses were weighted to produce the corresponding digital twin response based on the weighted average formulas given in equations (11) and (12). This process ensured that the models, and digital twin response, were continually updated to reflect the system's current state. The parameters of the physics-based model component, $\boldsymbol{M}_2$, are shown in Table 1 for each of the eight segments and Table 2 shows the evolution of the $\theta_2^{(5)}$ (overflow) parameter over time.

It is interesting to compare the overflow parameter values to the input signal throughout the segments. The fifth parameter was used in the model to allow for improved modelling of the overflow effects, which would occur under larger inputs where more water was pumped into the tanks. The input value reached its largest peaks in Segments 2 and 7, and this was when overflow was likely to occur. Over these segments, the overflow parameter was given the greatest values.

In fact, the overflow effects can be clearly seen in the water level of the second tank, which is shown in Fig 3 in comparison with the levels estimated by the two model components and the digital twin response. Overflow does occur in the second segment, as well as on the cusp of the seventh and eighth segments, and is marked by two flattened peaks in the measured response. Although the physics-based model's computed response ($\boldsymbol{M}_2$) is generally a close resemblance to the validation tank level, the NARX data-based model ($\boldsymbol{M}_3$) is clearly a much better fit with the

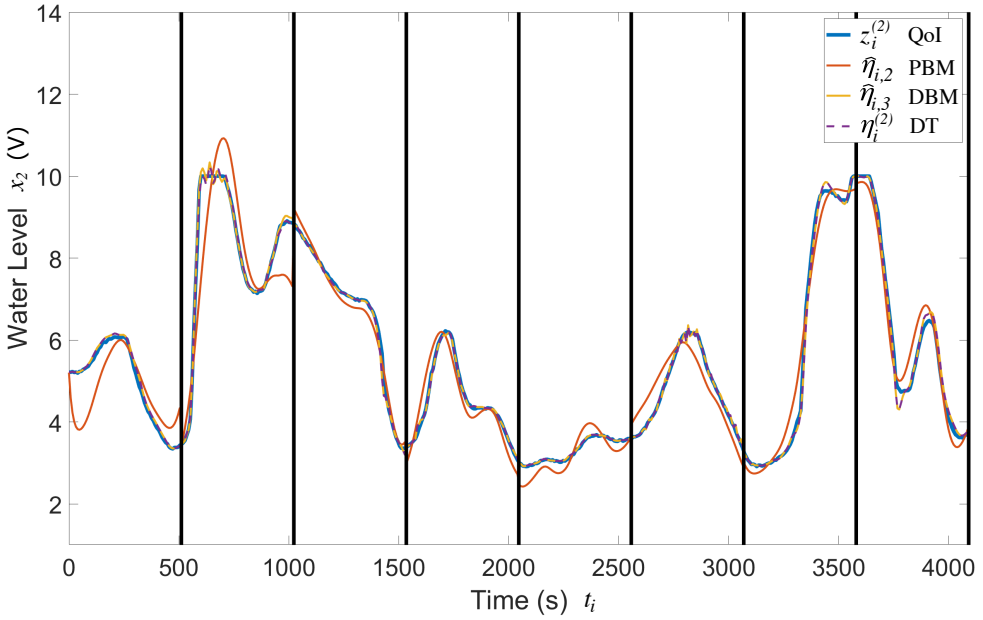

Figure 3: Comparison of the second tank water levels, x_2, computed using the physics-based model (PBM) M_2 (shown by $\hat{\eta}_{i,2}$) & data-based model (DBM) M_3 (shown by $\hat{\eta}_{i,3}$) and digital twin output function for $n = 2$ (denoted $\eta_i^{(2)}$ and DT) compared with the measured water level QoI $z_i^{(2)}$. Note that following [18] the units of x_2 are in volts (V) from the measurement.

Table 4
Mean-square error of the validation second tank water level computed using the physics-based model (PBM) & data-based model (DBM) components and digital twin combination.

Segment	1	2	3	4	5	6	7	8
MSE PBM	0.4787	0.9445	0.0420	0.1146	0.0776	0.2117	0.1906	0.1465
MSE DBM	0.0057	0.0075	0.0063	0.0016	0.0007	0.0071	0.0104	0.0103
MSE DT	0.0054	0.0075	0.0063	0.0016	0.0007	0.0071	0.0104	0.0103

QoI. This is particularly the case over the first two segments. The initial conditions of the system were unknown and affected the physics-based model's predictions over the first segment, and although the overflow parameter improved the modelling of the first overflow occurrence, there was clearly some shortfall in the model's capability.

It is unsurprising, given how much better the NARX model's match with the data were, that the model fractions were consistently less than 0.5 (and the updated model fraction equal to zero in all but one segment). Table 3 shows the physics-to-data model fractions used in the digital twin combination, both as originally calculated in equation (11) and as updated in equation (12). As a result, the digital twin most often assigned all weight to the data-based model's response in order to optimally match the tank level data.

Table 4 shows the mean-square error of the validation water levels of the second tank computed by each model component and the digital twin over time. The shortfall of the physics-based model component in modelling the overflow is evident in the mean-square error of its estimations in segments of larger inputs. In the second segment, which sees the largest input, the mean-square error is greatest, as seen in Table 4. On the other hand, where the input is smallest, the middle segments contain the best physics-based model estimations. While the data-based model (and digital twin) do appear to be affected by the tank overflow, the effects are much less noticeable and these models consistently offer significantly improved accuracy.

In the case of this choice of models and weighting functions, the physics-based model (M_2), although a good fit to the data, was poor in comparison with the data-based model (M_3). As a result, the model fraction was updated to zero over all but the second time segment, most often giving the physics-based model no weight in its contribution to the

digital twin. The resulting digital twin response was accurate and was easily able to reflect any changes in the tank's water level because of overflow effects when they occurred. An alternative weighting scenario is considered next.

3.4. Error-based weightings 2 — self-adaptive differential evolution

In this Section an ABC algorithm coupled with an efficient sampling technique called ABC-nested sampling (ABC-NS) is used to make inference. In the ABC-NS algorithm, the first iteration is similar to the ABC rejection-sampler. For the subsequent iteration, the tolerance threshold is defined based on the discrepancy values ranked in descending order as the $(\alpha_0 N)^{\text{th}}$ value where α_0 is equal to 0.8. The dropped particles represent 80 per cent from the total number of particles. After that, we normalise the weights of the remaining particles and a weight of zero is assigned to the dropped particles. From the remaining particles, a number of $(\beta_0 N)$ are randomly selected and propagated to the next population (β_0 is equal to 0.1). In this way, the algorithm can visit any part of the parameter space to ensure a good exploration. The remaining particles are then enclosed in an ellipsoid in which the mass center and the covariance matrix are estimated based only on the remaining particles. The aim of the elliptical bound is to restrict the parameter space around the most interesting part of the parameter space which improves the acceptance ratio and efficiency through the iterations. The volume of the generated ellipsoid could be enlarged by a factor f_0 to ensure that the particles on the borders will be inside. In this work f_0 is set to 1. Finally, the population is replenished by re-sampling particles inside the ellipsoid following the scheme in [20]. In the subsequent steps, the threshold is updated adaptively in the same way and samples selection are subjected to more stringent threshold. Through the populations and as $\epsilon_t \to 0$, a larger number of particles are selected and the samples for the parameters better reflect the real posterior distribution. A detailed discussion concerning the effects of these settings can be found in [1].

The physics-based model is chosen to be $\boldsymbol{M}_1$ (equation 14), and in order to estimate $\boldsymbol{\theta}_1$, the approximate Bayesian computation (ABC) nested sampling (NS) technique was used. As before the ABC method works as a proxy for the likelihood function by sampling a set of parameters and simulating data, using the resulting model. The selected physics-based model (PBM), $\boldsymbol{M}_1$, is deliberately chosen because it does not include the physics governing the overflow effects. This will be treated as additional model inadequacy, and will enable us to contrast the results with those, where $\boldsymbol{M}_2$ was used above. As a result, the statistical model for $n = 2$ is now given by

$$z_i^{(2)} = \rho_{s,1}\hat{\eta}_{i,1}^{(2)} + \rho_{s,3}\hat{\eta}_{i,3}^{(2)} + \delta_i + e_i = \eta_i^{(2)} + \delta_i + e_i \tag{18}$$

where δ_i represents the unknown combined model inadequacies, and $\eta_i^{(2)} = \rho_{s,1}\hat{\eta}_{i,1}^{(2)} + \rho_{s,3}\hat{\eta}_{i,3}^{(2)}$ is the combined digital twin output function. Other parameters are as previously defined.

The ABC-NS method [1] improves on the computational efficiency of other ABC algorithms by increasing the acceptance rate of proposed parameter values. The prior distributions chosen for the 4 parameters in $\boldsymbol{\theta}_1$ were informed by first running ABC-NS on the second data set (for prior knowledge) using the uniform distribution $\mathcal{U}[0, 0.1]$ for each parameter. The minimum and maximum values of the resulting final population determined the following prior distributions for $\boldsymbol{M}_1$:

$$\boldsymbol{\pi}(\theta^{(1)}) \sim \mathcal{U}[0.0389, 0.0893]$$
$$\boldsymbol{\pi}(\theta^{(2)}) \sim \mathcal{U}[0.0321, 0.0675]$$
$$\boldsymbol{\pi}(\theta^{(3)}) \sim \mathcal{U}[0.0271, 0.0465]$$
$$\boldsymbol{\pi}(\theta^{(4)}) \sim \mathcal{U}[0.0188, 0.0919].$$

The hyperparameters chosen for the ABC-NS (and used within all data segments) were enlargement factor $f_0 = 1.0$, dropped particle proportion $\alpha_0 = 0.8$, surviving particle proportion $\beta_0 = 0.1$, initial tolerance threshold $\epsilon_{tol3} = 100$, population size $N = 1000$ and convergence accuracy 0.1. All the hyperparameters are included in $\boldsymbol{\psi}_3$ for model $\boldsymbol{M}_3$. Upon convergence of the tolerance threshold, the mean of the resulting posterior distribution for each parameter was taken as its value in $\boldsymbol{M}_1$. The data-based model (DBM) was the same as previously, and is denoted $\boldsymbol{M}_3$ as before.

The responses of the PBM ($\boldsymbol{M}_1$) and DBM ($\boldsymbol{M}_3$) over the 8 data segments are compared to the QoI data in Fig 4. It is clear the DBM output (shown by $\hat{\eta}_{i,3}$) is consistently a better fit than the PBM (shown by $\hat{\eta}_{i,1}$) when compared to the QoI ($z_i^{(2)}$), particularly at higher water levels. The selected PBM, $\boldsymbol{M}_1$, does not include the physics governing the overflow effects, which occur in data segments 2 and 7. When compared with the previous results shown in Figure 3, where these effects were included in the PBM there does not appear to be significant differences. This may not be the case for other parameter values, particularly if the overflow effects lasted a longer time.

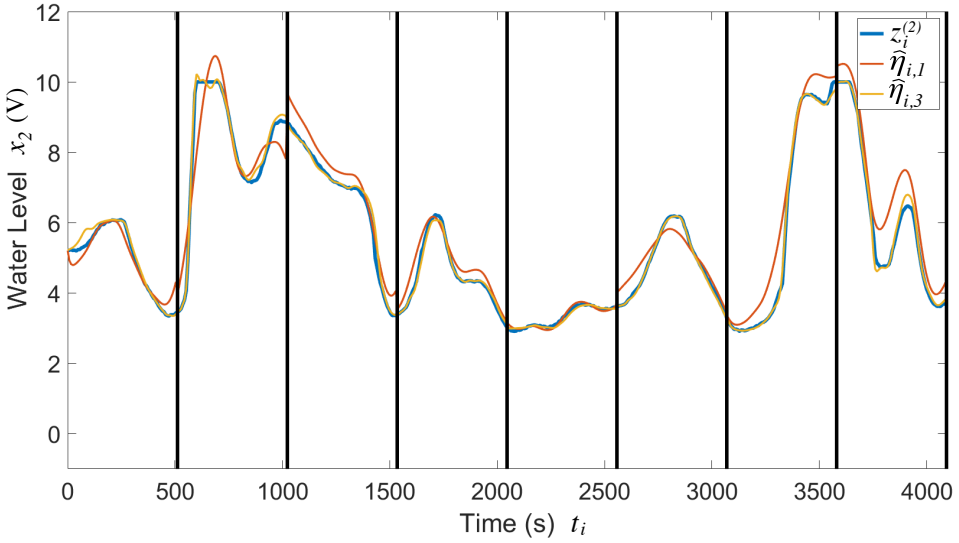

Figure 4: Water level, x_2, computed using M_1 (shown by $\hat{\eta}_{i,1}$) and M_3 (shown by $\hat{\eta}_{i,3}$) compared to the QoI water level, $z_i^{(2)}$ over the 8 data segments.

Table 5
The error-based physics-to-data model fraction over the 8 segments

Segment, s	1	2	3	4	5	6	7	8
$\Upsilon^e_{s,n}$	0.304	0.093	0.095	0.090	0.432	0.034	0.012	0.000

In Section 2.3, both error- and robustness-based optimisation criteria were described. Here the minimisation of NMSE is considered first; the optimal PBM and DBM error-based weights, $\rho^e_{s,1}$ and $\rho^e_{s,2}$ respectively, are defined as those which produce the combination of PBM and DBM outputs with minimum NMSE. To determine these error-based weights, a self-adaptive differential evolution (SADE) genetic algorithm was used [16, 22]. SADE optimises by proposing potential solutions, comparing them to an initial population of randomly-sampled values and keeping only the best ones [22].

The SADE algorithm used the following parameters: 3 runs, 20 maximum generations, population size 200, crossover ratio 0.5, scaling factor 1.5 and initial ranges [0, 1], and as before all these hyperparameters are stored in χ. The error-based physics-to-data model fraction, $\Upsilon^e_{s,n}$, for each of the 8 segments is shown in Table 5 (note that because there are just two combined models $\rho^e_{s,1} = \Upsilon^e_{s,n}$ and $\rho^e_2 = 1 - \Upsilon^e_{s,n}$). The values of the error-based model fraction reflect the fact that the DBM's estimation of the water level is often significantly more accurate (where a small portion of the total weight is assigned to the PBM). In fact within segment 8 the model fraction takes a value of 0, indicating the PBM's response cannot improve the DBM's in any way. On the other hand, the model fraction approaches closest to 0.5 (where the components' weights would be equal) in segments 1 and 5, where the DBM(PBM) is poorer (better) than in other segments.

3.4.1. Robustness-based weightings

The second optimisation criteria considered is the maximisation of robustness. The definition used here means that the optimal robustness-based weights, $\rho^r_{s,1}$ and $\rho^r_{s,2}$, are those which produce the combination of *worst-case* PBM & DBM outputs with minimum NMSE. While the models have point-value parameters, there is some uncertainty in their values (the ABC-NS resulted in distributions for each parameter – only the mean was selected – and repeatedly training the NN does not produce identical parameters each time). Parameter uncertainty then translates into uncertainty in the models' outputs, so there will be a worst-case prediction for each.

Table 6
The PBM parameters and worst-case parameters

Segment, s	1	2	3	4	5	6	7	8
Parameter Value								
k_1	0.0866	0.0880	0.0736	0.0511	0.0404	0.0474	0.0680	0.0611
k_2	0.0652	0.0655	0.0534	0.0389	0.0332	0.0365	0.0492	0.0447
k_3	0.0310	0.0296	0.0312	0.0434	0.0281	0.0419	0.0444	0.0456
k_4	0.0356	0.0319	0.0382	0.0472	0.0270	0.0449	0.0536	0.0557
Worst-Case Value								
k_1	0.0885	0.0889	0.0868	0.0398	0.0427	0.0406	0.0478	0.0432
k_2	0.0668	0.0635	0.0417	0.0435	0.0337	0.0426	0.0661	0.0529
k_3	0.0317	0.0278	0.0329	0.0417	0.0311	0.0377	0.0454	0.0455
k_4	0.0321	0.0298	0.0220	0.0680	0.0249	0.0595	0.0812	0.0846

Table 7
NMSE of the PBM and the worst-case prediction over the 8 segments.

Segment, s	1	2	3	4	5	6	7	8
Model NMSE	7.82	15.55	8.09	13.71	4.69	15.17	5.26	13.89
Worst-Case NMSE	1951	1306	1640	36833	2638	13712	4446	15601

3.4.2. PBM Worst-case prediction

As mini-max is a non-probabilistic approach to decision-making, no distributions are required for the model parameters. Instead, the minimum and maximum values in the last population returned by ABC-NS were used to inform a range of possible values for each parameter. Determining the worst-case PBM prediction, given these parameter ranges, requires an optimisation of its own. SADE was used again for this purpose, but with 30 maximum generations and population size 200. The parameters of the PBM and the values which produced the worst-case response are shown in Table 6. By re-calibrating the PBM with each new segment, the parameters are allowed to vary to give the best water level prediction with up-to-date data. Parameters k_1 and $k_2 \in \boldsymbol{\theta}_1$ appear to increase over segments with higher water levels. As may be expected, the parameters which produced the worst-case prediction were often equal to the limits of the uncertainty ranges. Table 7 shows how the NMSE of the worst-case prediction compares to that of the PBM with the assumed parameters. The segments within which the PBM was least robust to parameter uncertainty were 4, followed by 8 and then 6.

3.4.3. DBM worst-case prediction

The ranges of the possible NARX-NN parameters were determined by splitting the training data into 10 subsets and retraining the NN with each one. The minimum and maximum values of each parameter were taken as the limits of the range. As with the PBM, a SADE algorithm was used to find the worst-case MPO prediction of the NARX-NN within these parameter ranges. It also used 30 maximum generations, but this time with a population of 50 to limit computational expense. The parameters (weights and biases) of the NARX-NN, $[a_1, a_2, ...a_{11}] \in \boldsymbol{\theta}_3$, and the values which produced the worst-case response are shown in Table 8. As with other “black-box" DBMs, it is not possible to give any physical meaning to these parameters and they vary significantly between segments. However, the important consideration is the variation in the parameters between subsets of training data which gave relatively small ranges for each parameter, from which the worst-case values were chosen.

Table 9 shows how the NMSE of the worst-case prediction compares to that of the DBM with the assumed parameters. Even the worst-case NMSE values were significantly less than those of the PBM’s worst predictions, showing that the DBM is more robust to parameter uncertainty. However, comparing proportionally, the worst-to-model NMSE of both PBM & DBM were similar in multiple segments.

3.4.4. Robustness-based prediction

The minimax approach taken in this work aims to combine the two models’ worst-case predictions into the digital twin output function with minimum error. This simply follows the error-based optimisation procedure, but using the

Table 8
The DBM parameters and worst-case parameters

Segment, s	1	2	3	4	5	6	7	8
Parameter Value								
a_1	-0.2086	-0.3977	-0.3060	0.5497	0.0953	0.1718	0.2209	0.1011
a_2	-0.1154	0.4338	0.3844	0.0885	0.8856	-0.1983	1.1987	0.3567
a_3	0.0657	-0.2261	0.0772	0.6923	-0.1395	0.3903	1.8377	0.6611
a_4	-0.9713	2.5406	-0.6108	0.5527	1.1915	0.5994	-0.0138	-0.4433
a_5	-0.9046	0.1463	0.1951	0.4369	0.0435	0.2332	-1.0055	0.1905
a_6	0.8578	1.6947	2.4329	-0.4513	-1.9932	0.2880	0.6671	-2.7985
a_7	-1.7625	-1.3642	2.1076	-1.3020	1.6718	1.6360	0.5860	0.4874
a_8	-0.2040	0.1477	0.9896	0.2482	-0.6022	0.6423	0.0683	-0.2581
a_9	-0.1609	0.3495	-0.0649	0.0832	0.2591	-0.8912	-3.5373	-0.9390
a_{10}	3.3003	-1.5058	3.1085	1.4767	1.2856	-0.2625	2.3672	-3.6492
a_{11}	0.7308	0.0362	-0.4395	-0.2118	-1.0231	0.3185	0.1567	0.0049
Worst-Case Value								
a_1	-0.2319	-0.3978	-0.3775	0.5497	0.0953	0.1671	0.2386	0.0936
a_2	-0.0990	0.4324	0.3637	0.0928	0.8856	-0.1996	1.1987	0.3727
a_3	0.0825	-0.2256	0.0414	0.6848	-0.1395	0.3866	1.8378	0.6784
a_4	-0.9497	2.5416	-0.5576	0.5527	1.1915	0.5977	-0.0138	-0.4175
a_5	-0.9063	0.1460	0.1782	0.4421	0.0435	0.2361	-1.0056	0.1839
a_6	0.8607	1.6946	2.5530	-0.4543	-1.9932	0.2913	0.6846	-2.7990
a_7	-1.7619	-1.3642	2.1106	-1.3020	1.6718	1.6384	0.6451	0.4960
a_8	-0.2071	0.1477	0.9946	0.2482	-0.6022	0.6439	0.0692	-0.2506
a_9	-0.1399	0.3492	-0.0194	0.0850	0.2591	-0.8927	-3.5383	-0.9290
a_{10}	3.3059	-1.5058	3.1081	1.4766	1.2856	-0.2616	2.3164	-3.6523
a_{11}	0.7248	0.0361	-0.4575	-0.2173	-1.0231	0.3239	0.1839	0.0017

Table 9
NMSE of the DBM and the worst-case prediction over the 8 segments.

Segment, s	1	2	3	4	5	6	7	8
Model NMSE	2.51	0.63	1.28	0.74	3.58	0.17	0.121	0.48
Worst-Case NMSE	318	209	404	22.1	3.58	0.27	541	0.45

Table 10
The robustness-based physics-to-data model fraction over the 8 segments

Segment, s	1	2	3	4	5	6	7	8	
$\Upsilon^r_{s,n}$	0.000	0.000	0.000	0.019	0.000	0.000	0.000	0.000	0.000

worst-case predictions in place of the model outputs. Fig 5 shows the worst-case predictions of the PBM and DBM compared to the validation water level. A quick inspection of these predictions demonstrates that their combination should be heavily weighted towards the DBM in order to minimise error, particularly over the final segments. The same weight-optimising SADE parameters as in Section 3.3 were used to find the combination of these two worst-case responses with minimum NMSE. The robustness-based physics-to-data model fraction, $\Upsilon^r_{s,n}$, for each of the 8 segments is shown in Table 10 (note that here $\rho^r_{s,1} = \Upsilon^r_{s,n}$ and $\rho^r_{s,2} = 1 - \Upsilon^r_{s,n}$). Unsurprisingly, the DBM is heavily weighted in the optimal combination; only in segment 4 does the PBM's weight take a non-zero value. These robustness-based model fractions suggest that if robustness of the digital twin is highly important to the user, the component weights should strongly favour the DBM.

3.4.5. Digital twin weighted predictions

Having computed the responses of the individual PBM and DBM, and weightings for both error and robustness choices, a combined error plus robustness weighting can then be produced. This combination would aim to a balance of minimisation of the NMSE whilst maximising the robustness of the resulting combination. As described in Section

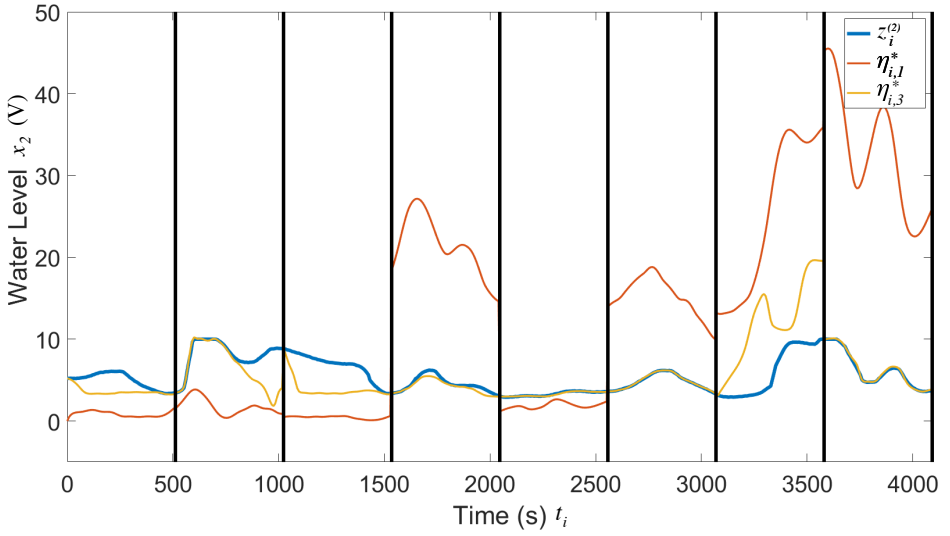

Figure 5: The worst-case predictions of the PBM, M_1, (shown by $\eta_{i,1}^*$) and DBM, M_3 (shown by $\eta_{i,3}^*$) compared to the QoI water level $z_i^{(2)}$ over the 8 data segments.

Table 11
The overall physics-to-data model fraction over the 8 segments

Segment, s	1	2	3	4	5	6	7	8
Υ	0.152	0.046	0.047	0.054	0.216	0.017	0.005	0.000

2.3 the overall weights of the two components' responses are determined as a linear combination of the error- and robustness-based weights. To demonstrate the concept, the value of α selected was 0.5, equally balancing NMSE and robustness equally. Based on this choice, the overall physics-to-data model fraction, $\Upsilon_{s,n}$, for each of the 8 segments is shown in Table 11 (where $\rho_{s,1} = \Upsilon_{s,n}$ and $\rho_{s,2} = 1 - \Upsilon_{s,n}$). Although the digital twin was already weighted in favour of the DBM over all 8 segments following the error-based optimisation, the robustness-based optimisation has further reduced the physics-to-data model fraction (essentially halving it in all segments with a robustness-based model fraction of 0). Segments 1 and 5 remain those with most influence from the PBM, and segment 8 still takes only the DBM response as the digital twin prediction.

Finally, Fig 6 shows the error- & robustness-based and overall weighted digital twin combinations compared to the QoI which is used as validation data. Note that these are all combinations of the model responses, not the worst-case predictions which were only used to determine the robustness-based model weights. The three combinations are all similar, but most difference occurs in the first two segments. The robustness-based weights give a slightly worse estimation of the QoI data than the NMSE-based (as would be expected). However, any error in the models' parameters would affect the estimated water level less. If the small level of error in the robustness-based combination is satisfactory for the user, it might be preferable to choose it over the more accurate error-based combination for this reason. With $\alpha = 0.5$, the overall digital twin combination is an even compromise between robustness and accuracy.

4. Discussion

The aim of this work was to produce digital twin outputs as a weighted combination of the responses of multiple model components. To enable the digital twin to evolve over time alongside its physical twin, the *digital twin output functions* are recalibrated for each segment of data recorded. This recalibration was based on computing weightings that were used in model combinations. Both error and robustness were considered as weighting criteria (property B1 from Section 2.2). The weighting choices made here were primarily made to demonstrate the concept, and in practice choices would need to be made relevant to the application being considered. Once the criteria for choosing weightings

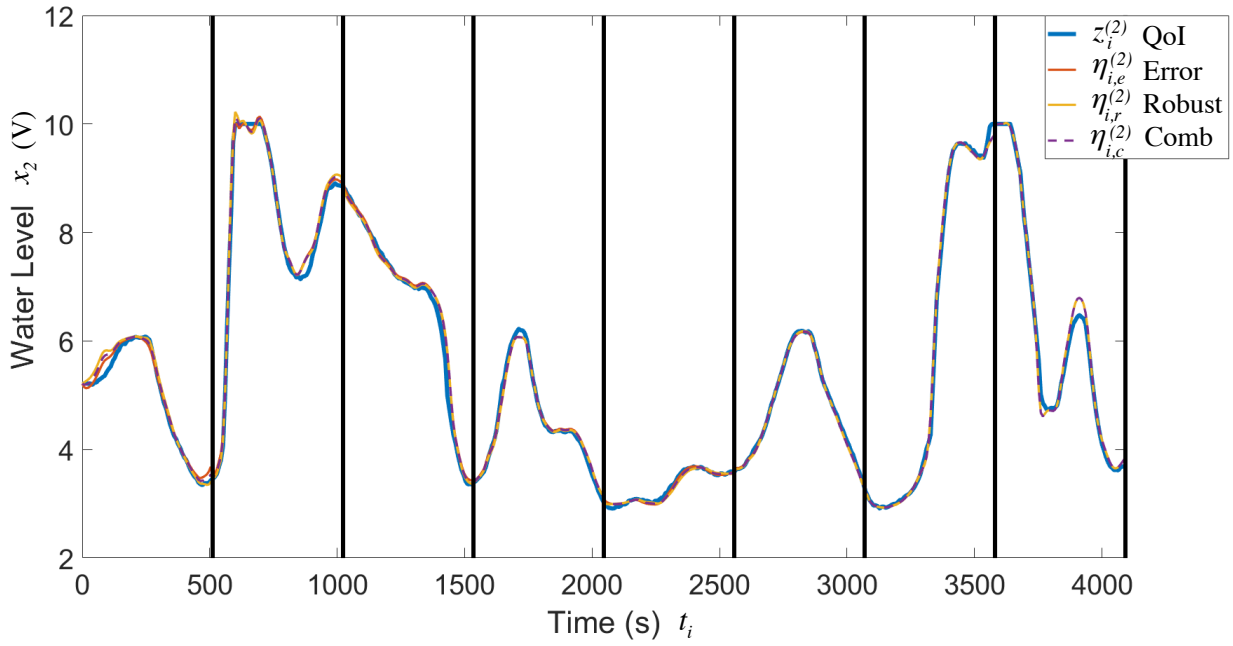

Figure 6: Comparison between the digital twin output functions and water level QoI $z_i^{(2)}$.Error-based weightings (orange line, denoted $\eta_{i,e}^{(2)}$ and 'Error'), robustness-based (yellow line, denoted $\eta_{i,r}^{(2)}$ and 'Robust') and combined error and robustness weighting (dashed purple line, denoted $\eta_{i,c}^{(2)}$ and 'Comb') compared to the measured water level data (blue line, labeled $z_i^{(2)}$ and 'QoI') over the 8 data segments.

were defined, predictions were then made based on weights computed using error- and robustness-based criteria, or some combination of both. In the example application in this paper, the decision to prioritise one optimisation criterion over the other makes little difference to the result, but this would not necessarily always be the case. Choosing α for the combined weighting would require a specific criteria to be defined by the user, based on the context of the application being considered.

The weighting design in this paper focused on property B1 (from Section 2.2) which is based on minimising error, as defined in Section 2.2. We did not consider properties B2 (predictive capability) or B3 (relative levels of deficiency) and because of this, the resulting weights tended to consistently (and often strongly) favour the DBM over the PBM. As a result, the physics-to-data model fractions that led to the best digital twin combinations to satisfy B1 were small (meaning that the DBM dominates). In terms of matching an output, DBMs have an advantage in that they are unconstrained by physics-based equations and aim only to learn the best input/output relationship. However, it is important to note that in this application the digital twin is only attempting to mirror the physical twin in its current state to ensure it is a good representation (interpolation), and not to make predictions about the system's future behaviour (extrapolation). While DBMs are typically unreliable outside the context in which they were trained, the physical constraints of PBMs mean they are generally much better at extrapolating to previously-unseen conditions. Therefore, when considering predictions of a system's hypothetical behaviour (i.e. property B2), the weighting choices would need to include criteria that reflect this aim.

Weighting choices for properties B2 and B3 is something we intend to consider in future work. In addition, probabilistic methods will be explored for dealing with the models' uncertainties, as an alternative to the minimax approach demonstrated in the current work. As ABC produces probability distributions for the PBM parameters, it would be sensible to incorporate Bayesian uncertainty analysis into the methodology, and probabilistic DBMs would be required in order to achieve this. Using this approach, the uncertainty of the digital twin output functions could be better quantified.

5. Conclusion

This paper presented a time-evolving digital twin concept, and applied it to an engineering dynamics example. The digital twin was developed by combining multiple physics-based and data-based computational models to create digital twin output functions. The model combination was optimised by weighting the responses of each model component based on minimisation of the NMSE and maximisation of the robustness to parameter uncertainty compared to the measured data. To ensure the digital twin can mirror a changing physical twin, the model components and weights are repeatedly re-calibrated using sequential segments of data measured from the physical twin. A physics-to-data model fraction was defined as a measure of the ratio of physics-based and data-based models for each digital twin output.

The method was applied to a cascaded tanks system to estimate the water level in one of the tanks over time. Approximate Bayesian computation was used to estimate the parameters of two physics-based models of the system, and a NARX neural network was trained as the data-based model. Examples were shown where weighted combinations of two (out of the three possible) models were selected based on their NMSE. The choice of weighting criteria meant that the data-based model was favoured over all data segments compared to the physics-based models. A similar trend was observed when the weights were chosen based on a robustness criteria, and the result showed that the data-based model was consistently strongly favoured. A final example demonstrated how the error- and robustness-based model weights could be combined, such that the resulting digital twin prediction of the tank's water level included criteria based on both accuracy and robustness to uncertainty.

Although the physics-based models were often assigned little (or zero) weight in the digital twin output function (the data-based model components were usually much more accurate), they would be important for extrapolating predictions in new contexts. Future work will focus on applying the digital twin to make such extrapolating predictions, as opposed to simply mirroring the physical system's current response, by adjusting the calculation of the physics-to-data model fraction, and therefore the weighting choice procedure.

CRediT authorship contribution statement

Lara Edington: Developing the concept, producing the simulations and writing of manuscript.. **Nikolaos Dervilis:** Developing the concept, and writing of manuscript. **Anis Ben Abdessalem:** Developing the ABC-NS concept, and writing of manuscript. **David Wagg:** Developing the concept, and writing of manuscript.

Funding Statement This work was funded by the University of Sheffield, Department of Mechanical Engineering ,departmental studentship for L.E and the EPSRC grant *Digital twins for improved dynamic design* (EP/R006768/1). For the purpose of open access, the author(s) has/have applied a Creative Commons Attribution (CC BY) licence to any Author Accepted Manuscript version arising

Data Availability Statement Replication data and code can be found in GitHub:
`\url{https://github.com/laraedington/digital twin}.`

Acknowledgements The authors would like to thank Keith Worden and Matthew Bonney for providing comments on the manuscript.

References

[1] Abdessalem, A.B., Dervilis, N., Wagg, D.J., Worden, K., 2019. Model selection and parameter estimation of dynamical systems using a novel variant of approximate Bayesian computation. Mechanical Systems and Signal Processing 122, 364–386.

[2] Ardabili, S., Mosavi, A., Várkonyi-Kóczy, A.R., 2019. Advances in machine learning modeling reviewing hybrid and ensemble methods, in: International Conference on Global Research and Education, Springer. pp. 215–227.

[3] Billings, S.A., 2013. Nonlinear system identification: NARMAX methods in the time, frequency, and spatio-temporal domains. John Wiley & Sons.

[4] Evensen, G., 2003. The ensemble Kalman filter: Theoretical formulation and practical implementation. Ocean dynamics 53, 343–367.

[5] Fuller, A., Fan, Z., Day, C., Barlow, C., 2020. Digital twin: Enabling technologies, challenges and open research. IEEE access 8, 108952–108971.

[6] Grieves, M., Vickers, J., 2017. Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems, in: Transdisciplinary perspectives on complex systems. Springer, pp. 85–113.

[7] He, B., Bai, K.J., 2021. Digital twin-based sustainable intelligent manufacturing: A review. Advances in Manufacturing 9, 1–21.

[8] Jones, D., Snider, C., Nassehi, A., Yon, J., Hicks, B., 2020. Characterising the digital twin: A systematic literature review. CIRP Journal of Manufacturing Science and Technology 29, 36–52.
[9] Kapteyn, M.G., Knezevic, D.J., Willcox, K., 2020. Toward predictive digital twins via component-based reduced-order models and interpretable machine learning, in: AIAA Scitech 2020 Forum, p. 0418.
[10] Kapteyn, M.G., Pretorius, J.V., Willcox, K.E., 2021. A probabilistic graphical model foundation for enabling predictive digital twins at scale. Nature Computational Science 1, 337–347.
[11] Kennedy, M.C., O'Hagan, A., 2001. Bayesian calibration of computer models. Journal of the Royal Statistical Society: Series B (Statistical Methodology) 63, 425–464.
[12] Liu, M., Fang, S., Dong, H., Xu, C., 2021. Review of digital twin about concepts, technologies, and industrial applications. Journal of Manufacturing Systems 58, 346–361.
[13] Minerva, R., Lee, G.M., Crespi, N., 2020. Digital twin in the IoT context: a survey on technical features, scenarios, and architectural models. Proceedings of the IEEE 108, 1785–1824.
[14] Niederer, S.A., Sacks, M.S., Girolami, M., Willcox, K., 2021. Scaling digital twins from the artisanal to the industrial. Nature Computational Science 1, 313–320.
[15] PTC, A., et al., 2005. Test uncertainty. American Society of Mechanical Engineers 19.
[16] Qin, A.K., Suganthan, P.N., 2005. Self-adaptive differential evolution algorithm for numerical optimization, in: Evolutionary Computation, 2005. The 2005 IEEE Congress on, IEEE. pp. 1785–1791.
[17] Särkkä, S., 2013. Bayesian filtering and smoothing. 3, Cambridge University Press.
[18] Schoukens, M., Mattson, P., Wigren, T., Noel, J.P., 2016. Cascaded tanks benchmark combining soft and hard nonlinearities, in: Workshop on nonlinear system identification benchmarks, pp. 20–23.
[19] Semeraro, C., Lezoche, M., Panetto, H., Dassisti, M., 2021. Digital twin paradigm: A systematic literature review. Computers in Industry 130, 103469.
[20] Shaw, J., Bridges, M., Hobson, M., 2007. Efficient bayesian inference for multimodal problems in cosmology. Monthly Notices of the Royal Astronomical Society 378, 1365–1370.
[21] Smith, R.C., 2013. Uncertainty quantification: theory, implementation, and applications. volume 12. Siam.
[22] Storn, R., Price, K., 1997. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. Journal of global optimization 11, 341–359.
[23] Sun, C., Shi, V.G., 2022. Physinet: A combination of physics-based model and neural network model for digital twins. International Journal of Intelligent Systems 37, 5443–5456.
[24] Tuegel, E.J., Ingraffea, A.R., Eason, T.G., Spottswood, S.M., 2011. Reengineering aircraft structural life prediction using a digital twin. International Journal of Aerospace Engineering 2011.
[25] Wagg, D., Gardner, P., Barthorpe, R., Worden, K., 2020a. On key technologies for realising digital twins for structural dynamics applications, in: Model Validation and Uncertainty Quantification, Volume 3. Springer, pp. 267–272.
[26] Wagg, D., Worden, K., Barthorpe, R., Gardner, P., 2020b. Digital twins: state-of-the-art and future directions for modeling and simulation in engineering dynamics applications. ASCE-ASME J Risk and Uncert in Engrg Sys Part B Mech Engrg 6.
[27] Ward, R., Choudhary, R., Gregory, A., Girolami, M., 2020. Continuous calibration of a digital twin: comparison of particle filter and Bayesian calibration approaches. arXiv preprint arXiv:2011.09810 .
[28] Worden, K., Barthorpe, R., Cross, E., Dervilis, N., Holmes, G., Manson, G., Rogers, T., 2018a. On evolutionary system identification with applications to nonlinear benchmarks. Mechanical Systems and Signal Processing 112, 194–232.
[29] Worden, K., Barthorpe, R.J., Cross, E.J., Dervilis, N., Holmes, G.R., Manson, G., Rogers, T.J., 2018b. On evolutionary system identification with applications to nonlinear benchmarks. Mechanical Systems and Signal Processing 112, 194–232.
[30] Worden, K., Cross, E., Gardner, P., Barthorpe, R., Wagg, D., 2020. On digital twins, mirrors and virtualisations, in: Model Validation and Uncertainty Quantification, Volume 3. Springer, pp. 285–295.
[31] Yamanaka, A., Maeda, Y., Sasaki, K., 2019. Ensemble Kalman filter-based data assimilation for three-dimensional multi-phase-field model: Estimation of anisotropic grain boundary properties. Materials & Design 165, 107577.
[32] Zhou, Z.H., 2019. Ensemble methods: foundations and algorithms. Chapman and Hall/CRC.