PHILOSOPHICAL TRANSACTIONS A

royalsocietypublishing.org/journal/rsta

Opinion piece

**Cite this article:** Worden K, Rogers T, Preston O. 2025 Machine-learning perspectives on Volterra system identification. *Phil. Trans. R. Soc. A* **383**: 20240053.
https://doi.org/10.1098/rsta.2024.0053

Received: 11 February 2025
Accepted: 20 May 2025

One contribution of 14 to a theme issue 'Frontiers of applied inverse problems in science and engineering'.

**Subject Areas:**
mechanical engineering

**Keywords:**
Volterra series, nonlinear dynamics, machine learning

**Author for correspondence:**
Keith Worden
e-mail: k.worden@sheffield.ac.uk

Supplementary material is available online at https://doi.org/10.6084/m9.figshare.c.8039942.

THE ROYAL SOCIETY PUBLISHING

# Machine-learning perspectives on Volterra system identification

Keith Worden, Timothy Rogers and Oliver Preston

Dynamics Research Group, School of Mechanical, Aerospace and Civil Engineering, The University of Sheffield, Sheffield S1 3JD, UK

KW, 0000-0002-1035-238X; TR, 0000-0002-3433-3247

The Volterra series has been used in nonlinear system identification (NLSI) for decades; its frequency-domain counterpart allows a generalization of 'resonance curves' for nonlinear systems—so-called higher-order frequency-response functions (HFRFs). Estimating the terms in the series has often proved to be a challenge; however, the (comparatively) recent uptake of machine-learning technology into engineering dynamics has led to advances in the identification of the series—both for the Volterra kernels themselves and for the HFRFs. The current paper provides an overview of a number of approaches based on neural networks, Gaussian processes (GPs) and reproducing kernel Hilbert spaces (RKHSs), and presents new results for multi-input multi-output (MIMO) systems based on neural networks.

This article is part of the theme issue 'Frontiers of applied inverse problems in science and engineering'.

## 1. Introduction

One of the most important inverse problems in structural dynamics is *system identification* (SI); this is the problem of fitting a mathematical model of a system to measured data. Even for linear systems, this is an inverse problem of the second kind and is very often ill-posed [1]. For nonlinear systems, the problem is much more difficult, as there are essentially an infinity of possible model forms to choose from. In general, there is no 'one-size-fits-all' solution to problems in nonlinear SI (NLSI), and the practitioner must often rely on a 'toolbox' approach, where different methods suit different problems [1]. In its most abstract framing, the system can be regarded mathematically as a functional $S$ which maps an input or stimulus function of time $x(t)$, into an output or response function $y(t)$; e.g. $y(t) = S[x(t)]$. In most cases, the functional is only given implicitly by an 'equation of motion'; for example, Newtonian physics of mechanical/structural systems often leads to second-order differential equations like,

$$m\ddot{y} + c\dot{y} + ky + k_3y^3 = x(t), \tag{1.1}$$

which is a form of Duffing's equation, which is of fundamental importance in nonlinear dynamics. If the form of the model is known, the identification problem is reduced to that of *parameter estimation*; i.e. the determination of some 'best' set of values for the parameters $\{m, c, k, k_3\}$. The problem thus becomes one of *regression* [2]. For many years, problems of this kind were approached using *least-squares-error* approaches of varying levels of sophistication. If the form of the model is not known *a priori*, the problem becomes much more difficult, as NLSI then requires a *structure detection* step before parameters can be estimated. A powerful alternative to parametric identification has emerged in the last few decades, based on ideas from the discipline of machine learning [2]. This approach is based on the idea of fitting a *non-parametric* or 'black-box' model to data.[1] Usually, the idea is to specify some mathematical basis of functions—which need not carry any clear physical meaning—which can serve as a *universal approximator*. One essentially fits a superposition of these basis functions to data from the system of interest to minimize an error function of some sort; the universal approximation property then ensures that models of arbitrarily high fidelity can be obtained, assuming that enough basis functions are used (this leaves aside the problem of noise on the data for the moment). One of the most well-known non-parametric model bases is provided by the *artificial neural network* paradigm [2], but there are many alternatives. In fact, non-parametric models have been around for a very long time, one of the earliest being the *Volterra series* [1,4,5].

The eponymous Volterra series was first proposed in 1887 [6]. Volterra's intention in the paper was to extend the theory of analytic functions to functionals; this he did by proposing a type of Taylor series appropriate to the more general object.[2] Despite its origins in pure mathematics, the Volterra series has proved to be useful in applied analysis. Although one of the main applications of the series was in the modelling of physiological systems [7], there was also substantial work in the non-parametric identification and modelling of nonlinear dynamic systems—which started earlier—largely with the seminal work of Barrett [8,9]. Barrett's work mainly found interest in the electrical and control engineering communities until it was adopted for structural SI in the work of Gifford and Tomlinson [10]. Apart from its potential for non-parametric identification, the Volterra approach generated interest in the structural dynamics community because of its ability to form nonlinear extensions of 'resonance curves'. These *higher-order frequency-response functions* (HFRFs) [1,11] can provide visual and interpretable representations of nonlinear dynamic systems and explain how input frequencies might combine to create nonlinear analogues of resonance conditions under conditions of weak nonlinearity.

Throughout the history of the Volterra series, one of the main problems has been that of computing or estimating the terms of the series. (It will soon become clear how demanding this problem is.) Various ingenious (direct) approaches have been proposed over the years [1]; however, the demands of algebra or data meant that any analysis was always confined to the lower-order terms of the series—usually the first three. More recently, approaches have arrived based on non-parametric machine learning. The first of these developments was that of Wray & Green [12], who proposed a means of estimating the series using a time-delay neural network. The method of Wray & Green provided a means of finding the kernels for a discrete Volterra series in terms of the network weights; this was followed by the study in [13] which, in contrast, used neural network weights to directly find the kernel transforms or HFRFs. A more recent development was the formulation of a *reproducing kernel Hilbert space* (RKHS) approach to the series, which in principle can estimate the entire series in one go [14–16]. Even more recently, a Gaussian process (GP) approach was presented in [17,18], which simplified and extended the work [13], but also provided a means or characterizing the uncertainty in HFRF estimates. The current paper takes the viewpoint that a *modern perspective* on the Volterra series is one in which one can exploit modern machine-learning methods to determine the series for a given system of interest; as such, it provides illustrations of the artificial neural network, RKHS and GP approaches to estimating kernels and kernel transforms. The aim of the paper is not to provide a survey or overview of methods; instead, it rather shamelessly draws on examples from the authors' previous work, together with some new results.

The layout of the paper is as follows. In §2, we introduce the main terminology and offer a demonstration that the Volterra series is indeed a functional Taylor series. In §3, we present the RKHS formulation of the series, which reduces the estimation of kernels to a type of regularized least-squares problem. In §4, we show how one can estimate HFRFs directly from nonlinear autoregressive exogenous (NARX) models and illustrate the idea using GP-NARX models, which also have the advantage of a Bayesian formulation. The NARX approach is extended to multi-input multi-output (MIMO) models in §5; this time using a neural network as the basis for the model. Some conclusions are drawn in the final section.

## 2. The Volterra series

It is basic knowledge in structural/engineering dynamics that linear systems admit dual time and frequency- domain characterizations,

$$y(t) = \int_{-\infty}^{\infty} \mathrm{d}\tau\, h(\tau)x(t-\tau) \tag{2.1}$$

and,

$$Y(\omega) = H(\omega)X(\omega). \tag{2.2}$$

All information about a *single-input-single-output* system is encoded in either the *impulse response function* $h(t)$ or the *frequency-response function* (FRF) $H(\omega)$. The best representation for a given situation will be often be problem-specific. For vibration problems, the frequency-domain approach is usually adopted; displaying the FRF $H(\omega)$ shows immediately those frequencies at which large outputs can be expected, i.e. peaks in $H(\omega)$ corresponding to the system resonances.

Equations (2.1) and (2.2) are manifestly linear and therefore cannot hold for arbitrary nonlinear systems; however, both admit a generalization based on the (infinite) Volterra series,

$$y(t) = y_1(t) + y_2(t) + y_3(t) + \dots, \tag{2.3}$$

where,

$$y_1(t) = \int_{-\infty}^{+\infty} \mathrm{d}\tau h_1(\tau)x(t-\tau), \tag{2.4}$$

$$y_2(t) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} \mathrm{d}\tau_1 \mathrm{d}\tau_2 h_2(\tau_1,\tau_2)x(t-\tau_1)x(t-\tau_2), \tag{2.5}$$

and so on; the form of the general term follows directly from the above. The functions $h_1(\tau)$, $h_2(\tau_1,\tau_2)$, $h_3(\tau_1,\tau_2,\tau_3)$, ..., $h_n(\tau_1,\dots,\tau_n)$, ... are generalizations of the linear impulse response function and are usually referred to as *Volterra kernels*; one can also think of them as the analogue of coefficients in a standard Taylor series. The series allows a representation of a functional,[3]

$$y(t) = V[x(t)]. \tag{2.6}$$

It is straightforward to show that the Volterra kernels can be taken to be completely symmetric in their arguments [1]. In this paper, the methods of extracting the kernels and their transforms directly will automatically select the symmetric kernels.

Now, the Volterra series has been referred to as a *functional Taylor series*; while it is clearly a power series in the $x(t)$, the term 'Taylor series' arguably requires further justification. In fact, although this justification is the object of the first Volterra paper [6], it is very rarely given, so it is interesting to provide it here. Of course, this is not a modern perspective by any means; however, it can be given a modern slant. An important tool can be applied which was unknown at the time of Volterra's; this is the *functional derivative*, defined as [20]:

$$\frac{\delta}{\delta f(y)}F(f) = \lim_{h\to 0}\frac{1}{h}(F[f(x) + h\delta(x-y)] - F[f(x)]), \tag{2.7}$$

where $\delta(x)$ is the *Dirac delta-function*. For the discussion here, it will be sufficient to consider the truncated functional

$$V[x(t)] = \int_{-\infty}^{+\infty} \mathrm{d}\tau h_1(\tau)x(t-\tau) + \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} \mathrm{d}\tau_1\mathrm{d}\tau_2 h_2(\tau_1,\tau_2)x(t-\tau_1)x(t-\tau_2). \tag{2.8}$$

Consider the first term, and make a simple change of variables, so that

$$y_1[x(t)] = \int_{-\infty}^{+\infty} \mathrm{d}\tau h_1(t-\tau)x(\tau). \tag{2.9}$$

On applying equation (2.7) and forming the first functional derivative, one sees that

$$\frac{\delta}{\delta x(T)}y_1[x(t)] = \lim_{h\to 0}\frac{1}{h}\{y_1[x(t) + h\delta(t-T)] - y_1[x(t)]\} \tag{2.10}$$

and substituting the form of $y_1[x(t)]$ from equation (2.9) gives,

$$\frac{\delta}{\delta x(T)}y_1[x(t)] = \lim_{h\to 0}\frac{1}{h}\left\{\int_{-\infty}^{+\infty} \mathrm{d}\tau h_1(t-\tau)[x(\tau) + h\delta(\tau-T)],\right.$$

$$\frac{\delta}{\delta x(T)}y_1[x(t)] = \lim_{h\to 0}\frac{1}{h}\left\{\int_{-\infty}^{+\infty} \mathrm{d}\tau h_1(t-\tau)[x(\tau) + h\delta(\tau-T)] - \int_{-\infty}^{+\infty} \mathrm{d}\tau h_1(t-\tau)x(\tau)\right\}. \tag{2.11}$$

The only awkward term in this expression is the one involving the delta function, as the other two cancel. Concentrating on the 'delta' term, one sees that,

$$\lim_{h\to 0}\frac{1}{h}\int_{-\infty}^{+\infty} \mathrm{d}\tau h_1(t-\tau)h\delta(\tau-T),$$

$$\int_{-\infty}^{+\infty} d\tau h_1(t-\tau)\delta(\tau-T) = h_1(t-T) = \frac{\delta y_1[x(t)]}{\delta x(T)}, \tag{2.12}$$

after observing that the $h$ cancels, so that the limit goes away, and then using the projection property of the delta function. The end result is,

$$y_1[x(t)] = \int_{-\infty}^{+\infty} d\tau \left.\frac{\delta y_1[x(t)]}{\delta x(T)}\right|_{T=\tau} x(\tau). \tag{2.13}$$

So, to the first order of truncation only,

$$y_1[x(t)] = \int_{-\infty}^{+\infty} d\tau \left.\frac{\delta V[x(t)]}{\delta x(T)}\right|_{T=\tau} x(\tau). \tag{2.14}$$

This is now suggestive of a Taylor series; there is a single power of $x(t)$, multiplied by the derivative of the functional $V$ with respect to $x(t)$. The other thing to note is that the evaluation at $T=\tau$ for the derivative is not quite correct; the functional does not depend on $T$ or $\tau$, it depends on the *whole function* $x(\tau)$. In fact, the functional derivative in equation (2.12) does not depend on $x(\tau)$ at all, so it does not matter where one evaluates it. This point will be important shortly.

The second integral in equation (2.8) requires a little more effort; one begins by taking a first functional derivative,

$$\frac{\delta y_2[x(t)]}{\delta x(T_1)} = \lim_{h\to 0}\frac{1}{h}\left\{\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} \mathrm{d}\tau_1\mathrm{d}\tau_2 h_2(t-\tau_1, t-\tau_2)[x(\tau_1)+h\delta(\tau_1-T_1)][x(\tau_2)+h\delta(\tau_2-T_1)]\right.$$
$$\left. -\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}\mathrm{d}\tau_1\mathrm{d}\tau_2 h_2(t-\tau_1,t-\tau_2)x(\tau_1)x(\tau_2)\right\}. \tag{2.15}$$

As before, the product of $x$'s in the first term cancels with the last term. A further simplification comes from the fact that the $O(h^2)$ term arising from a product $[h\delta(\tau_1-T_1)][h\delta(\tau_2-T_1)]$ vanishes in the limit that $h\longrightarrow 0$. Finally, the $O(h)$ terms are multiplied by $1/h$, and the limit goes away as before. All that remains is

$$\begin{aligned}\frac{\delta y_2[x(t)]}{\delta x(T_1)} &= \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}\mathrm{d}\tau_1\mathrm{d}\tau_2 h_2(t-\tau_1,t-\tau_2)\{x(\tau_1)\delta(\tau_2-T_1)+x(\tau_2)\delta(\tau_1-T_1)\}\\ &= \int_{-\infty}^{+\infty}\mathrm{d}\tau_1 h_2(t-\tau_1,t-T_1)x(\tau_1)+\int_{-\infty}^{+\infty}\mathrm{d}\tau_2 h_2(t-T_1,t-\tau_2)x(\tau_2).\end{aligned} \tag{2.16}$$

Finally, changing the 'dummy' variable $\tau_2$ in the second integral to $\tau_1$ and using the symmetry of the $h_2$ gives

$$\frac{\delta y_2[x(t)]}{\delta x(T_1)} = 2\int_{-\infty}^{+\infty}\mathrm{d}\tau_1 h_2(t-\tau_1,t-T_1)x(\tau_1). \tag{2.17}$$

Now, taking the second derivative is basically the same as taking the first derivative of $y_1[x(t)]$, and the result is,

$$\frac{\delta}{\delta x(T_2)}\frac{\delta y_2[x(t)]}{\delta x(T_1)} = \frac{1}{2}h_2(t-T_2,t-T_1), \tag{2.18}$$

so the corresponding term in the Taylor series for $V[x(t)]$ is

$$\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}\mathrm{d}\tau_1\mathrm{d}\tau_2\frac{1}{2!}\left.\frac{\delta^2 V[x(t)]}{\delta x(T_2)\delta x(T_1)}\right|_{?} x(\tau_1)x(\tau_2), \tag{2.19}$$

which has exactly the form one might expect for the second term of a functional Taylor series. The only outstanding question is of where the functional derivative should be evaluated. This can now be established. The analysis of the second-order term has shown that the nice clean result obtained for equation (2.13) was just because the derivative was only of $y_1[x(t)]$; in fact taking the first derivative of $V[x(t)]\approx y_1[x(t)]+y_2[x(t)]$ yields,

$$\frac{\delta V[x(t)]}{\delta x(T)} = h_1(t-T)+2\int_{-\infty}^{+\infty}\mathrm{d}\tau h_2(t-\tau,t-T)x(\tau), \tag{2.20}$$

and the second term is unwanted. However, it can be removed by simply evaluating the functional at the zero-function $x(t)=0$ for all $t$, so,

$$\left.\frac{\delta V[x(t)]}{\delta x(T)}\right|_{x(T)=0} = h_1(t-T), \tag{2.21}$$

and this is clearly the correct prescription for a Taylor-type series (strictly a Maclaurin-type series). Continuing the analysis of the second derivative to $y_3[x(t)]$ would also yield a linear functional in

$x(t)$, so a clean result for the second derivative requires that it too is evaluated at $x(t) = 0$, and so on. To second order, the final result is,

$$V[x(t)] = \int_{-\infty}^{+\infty} \mathrm{d}\tau \left.\frac{\delta V[x(t)]}{\delta x(T)}\right|_{x(T)=0} x(\tau) + \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} d\tau_1 \mathrm{d}\tau_2 \frac{1}{2!} \left.\frac{\delta^2 V[x(t)]}{\delta x(T_2)\delta x(T_1)}\right|_{x(T)=0} x(\tau_1)x(\tau_2) + O(x^3), \tag{2.22}$$

and it is shown that the Volterra series is indeed a functional Taylor series with 'coefficients' equal to the Volterra kernels. Furthermore, the analysis shows—in a non-rigorous fashion admittedly—that the existence of the Volterra series is conditional on the existence of the functional derivatives of all orders; this shows that the series is only defined for 'smooth' functionals. A more rigorous discussion of validity of the series can be found in [19].

The analysis so far has been confined to the time domain; in fact, as stated above, there exists a dual frequency-domain representation for nonlinear systems. The *higher-order FRFs* or *Volterra kernel transforms* $H_n(\omega_1, \dots, \omega_n)$, $n = 1, \dots, \infty$ are defined as the multidimensional Fourier transforms of the time-domain kernels, i.e.,

$$H_n(\omega_1, \dots, \omega_n) = \int_{-\infty}^{+\infty} \dots \int_{-\infty}^{+\infty} \mathrm{d}\tau_1 \dots \mathrm{d}\tau_n h_n(\tau_1, \dots, \tau_n) e^{-i(\omega_1\tau_1 + \dots + \omega_n\tau_n)} \tag{2.23}$$

,

$$h_n(\tau_1, \dots, \tau_n) = \frac{1}{(2\pi)^n} \int_{-\infty}^{+\infty} \dots \int_{-\infty}^{+\infty} \mathrm{d}\omega_1 \dots \mathrm{d}\omega_n H_n(\omega_1, \dots, \omega_n) e^{+i(\omega_1\tau_1 + \dots + \omega_n\tau_n)} \tag{2.24}$$

.

It is a simple matter to show that symmetry of the kernels implies symmetry of the kernel transforms; so for example, $H_2(\omega_1, \omega_2) = H_2(\omega_2, \omega_1)$.

The analysis to obtain the frequency-domain dual of equation (2.3) is quite straightforward [1]; the result is

$$Y(\omega) = Y_1(\omega) + Y_2(\omega) + Y_3(\omega) + \cdots, \tag{2.25}$$

where,

$$Y_1(\omega) = H_1(\omega)X(\omega), \tag{2.26}$$

$$Y_2(\omega) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} d\omega_1 H_2(\omega_1, \omega - \omega_1) X(\omega_1) X(\omega - \omega_1), \tag{2.27}$$

and so on. As mentioned earlier, the fundamental problem associated with the Volterra series is the determination of either the kernels or the kernel transforms. This process must be done analytically, if the equations of motion are known, or numerically, if time-series are given for the input and output processes. In the next section, analysis using RKHS ideas will be discussed.

## 3. The reproducing kernel Hilbert space Volterra series

The analysis here will follow the original work in [14–16]; for brevity, a great deal of the basic theory of RKHSs will be omitted, the reader should consult [21,22].

The first step in constructing a kernel version of the Volterra series is to move to a discrete form of the expansion; the argument here will proceed via a discussion of time-series models, as other forms of such models will be considered in detail later. As above, the input signal for the model will be denoted by $x(t)$ and the output by $y(t)$. The relevant time-series form for the Volterra series is

$$y_i = f(x_i, x_{i-1}, x_{i-2}, \ldots), \tag{3.1}$$

where the $x_i$ and $y_i$ are values of $x(t)$ and $y(t)$ sampled at a discrete set of times $\{t_i\}$. It will be assumed that the sample times are equally spaced, with the *sampling interval* denoted by $\Delta t$. Thus, $x_i = x(t_0 + i\Delta t)$, and similarly for $y_i$; for simplicity and without loss of generality, the start time $t_0$ will be taken as $t_0 = 0$. With these conventions, the *lagged variables* are given by $x_{i-n} = x(t_i - n\Delta t)$ etc. One then sees that the model form in equation (3.1) simply regresses the current output on past values of the input, where $f$ is some nonlinear multivariate function of choice. In the time-series literature, a model of this form is called an *NX-model* (nonlinear with exogenous inputs). Now, as long as $f$ is reasonably well-behaved, one can expand it as a multivariate Taylor series with polymonial terms. The result is then an infinite series,

$$y_i = h_0 + \sum_{j=0}^{\infty} h_1^j x_{i-j} + \sum_{j_1=0}^{\infty}\sum_{j_2=0}^{\infty} h_2^{j_1 j_2} x_{i-j_1} x_{i-j_2} + \ldots, \tag{3.2}$$

and this is the discrete Volterra series. The 'kernels' in this expansion are not functions, but vectors, matrices, and so on, $h_1^j$, $h_2^{j_1,j_2}$, indexed by the order of nonlinearity and the relevant lag values. Note that forward 'lags' are ruled out by the demands of causality in the NX model—the output cannot depend on inputs in the future.

In the discrete case, the coefficients in the expansion are ordinary partial derivatives rather than functional derivatives, i.e.

$$h_1^j = \frac{\partial f}{\partial x_{i-j}} \tag{3.3}$$

and so on. By the same argument as in the previous section, one can assume that the discrete kernels $h_n^{j_1,\ldots,j_n}$ are totally symmetric on their indices.

Formally, equation (3.2) is an infinite series; however, the identification problem has become that of determining the series coefficients, so these will need to be reduced to a finite set. The full series is usually truncated in two ways. In the first place, the polynomial order of the expansion is bounded at some value $L$; i.e. only the first $L$ discrete kernels are estimated. Second, a maximum lag $M$ is fixed; one can think of this as the *memory* of the series. With these constraints, the series becomes,

$$y_i = h_0 + \sum_{j=0}^{M-1} h_1^j x_{i-j} + \sum_{j_1=0}^{M-1}\sum_{j_2=0}^{M-1} h_2^{j_1 j_2} x_{i-j_1} x_{i-j_2} + \ldots + \sum_{j_1=0}^{M-1} \cdots \sum_{j_L=0}^{M-1} h_L^{j_1,\ldots,j_L} x_{i-j_1} \ldots x_{i-j_L}, \tag{3.4}$$

or, in a much more compact (but less transparent) form,

$$y_i = h_0 + \sum_{l=1}^{L} \left\{ \sum_{j_1=0}^{M-1} \cdots \sum_{j_l=0}^{M-1} h^l_{j_1,\ldots,j_l} \prod_{k=1}^{l} x_{i-j_k} \right\}. \tag{3.5}$$

Assuming enough training data (there are potentially many coefficients here), one could frame the identification as a (potentially very large), least-squares estimation problem for the parameters $h_1^j$, $h_2^{j_1 j_2}$ etc. However, the analysis will proceed here based on Dodd & Harrison's elegant RKHS approach [14–16].

The first step will be to construct an RKHS appropriate to the problem. It will be necessary to develop a little more notation.

First, one defines an input vector $\underline{z}$=$(x_i, x_{i-1}, \dots, x_{i-M+1})$. To avoid a multiplicity of indices, this vector will usually be taken to refer to the current instant $t_i$, so that,

$$y_i = h_0 + \sum_{l=1}^{L} \left\{ \sum_{j_1=0}^{M-1} \cdots \sum_{j_l=0}^{M-1} h^l_{j_1,\dots,j_l} \prod_{k=1}^{l} z_k \right\}. \tag{3.6}$$

This form is just a large polynomial expansion in $z_k$; for example, if $L=2$ and $M=2$, the expansion basis for the model is simply $\underline{\phi}(\underline{z}) = (1, z_1, z_2, z_1 z_2, z_1^2, z_2^2) = (1, x_i, x_{i-1}, x_i x_{i-1}, x_i^2, x_{i-1}^2)$. Here, $\underline{\phi}(\underline{z})$ represent the expansion in terms of feature-space embedding; this is the first hint of the RKHS approach. It is fairly straightforward to show that the number of terms in the expansion, and thus the dimension of the embedding space is $V_n = (L+M)!/L!M!$.

After a certain amount of straightforward but tedious algebra,[4] the model form equation (3.6) is converted to

$$y_i = \sum_{k=1}^{V_n} w_k \phi_k(\underline{z}), \tag{3.7}$$

where $w_k$ are just the various Volterra coefficients in appropriate order, starting from $w_1 = h_0$.

Note that this is a standard RKHS embedding via $\underline{\phi}$, from the measured-data space into a high-dimensional feature space, which is going to be the RKHS space: i.e., $\underline{\phi}: \mathbb{R}^M \longrightarrow \mathcal{H}$. For the example immediately above, one has $M=2$ and $V_n = 6$. Of course, one needs to be sure that $\mathcal{H}$ is indeed a complete inner-product space.

For general $L$, $M$, and thus $V_n$, one defines $\mathcal{H}$ to be the space of functions of the form

$$y_i = \sum_{k=1}^{V_n} w_k \phi_k(\underline{z}) = < \underline{w}, \underline{\phi} >_{\mathbb{R}^{V_n}} \tag{3.8}$$

for arbitrary $w_k \in \mathbb{R}$, and formally allow $V_n$ to be infinite if necessary.

Now, one defines the inner product on $\mathcal{H}$ to be

$$\left\langle \sum_{k=1}^{V_n} w_k \phi_k(\underline{z}), \sum_{k=1}^{V_n} v_k \phi_k(\underline{z}) \right\rangle_{\mathcal{H}} = \sum_{k=1}^{V_n} \frac{w_k v_k}{\lambda_k}, \tag{3.9}$$

where the $\lambda_k$ are chosen so that the series on the r.h.s. converges. (Of course, convergence is not an issue if $V_n$ is finite.)

This prescription, the expansion in equation (3.8), together with the inner product in equation (3.9) specifies a Hilbert space $\mathcal{H}$. However, it is still not an RKHS. First, one needs a kernel; this is defined here as

$$k(\underline{z}, \underline{z}') = \sum_{k=1}^{V_n} \lambda_k \phi_k(\underline{z}) \phi_k(\underline{z}'). \tag{3.10}$$

The question now is whether the Hilbert space $\mathcal{H}$ is an RKHS; one must check two important technical properties.

(1) First, that for a fixed $\underline{z}$, one has $k(\underline{z}, .) \in \mathcal{H}$. This is clear as,

$$k(\underline{z}, .) = \sum_{k=1}^{V_n} \lambda_k \phi_k(\underline{z}) \phi_k(.), \tag{3.11}$$

and for a given $\underline{z}$, the $\lambda_k \phi_k(\underline{z})$ are just numbers, say $w'_k$. One thus has,

$$k(\underline{z}, .) = \sum_{k=1}^{V_n} w'_k \phi_k(.) \in \mathcal{H}, \tag{3.12}$$

which has the required form from equation (3.8).

(2) Second, one must have the reproducing property. For a given $y \in \mathcal{H}$, one requires [23],

$$< y, k(\underline{z},.) >= y(\underline{z}). \tag{3.13}$$

This is immediate as,

$$y = \sum_{k=1}^{V_n} w_k \phi_k(.), \tag{3.14}$$

so that,

$$< y, k(\underline{z},.) >= \sum_{k=1}^{V_n} \frac{w_k w'_k}{\lambda_k} = \sum_{k=1}^{V_n} \frac{w_k \lambda_k \phi(\underline{z})}{\lambda_k} = \sum_{k=1}^{V_n} w_k \phi_k(\underline{z}) = y(\underline{z}). \tag{3.15}$$

The point now is that expansions such as

$$y(\underline{z}) = \sum_{k=1}^{V_n} w_k \phi_k(\underline{z}), \tag{3.16}$$

which might involve a very large (or even infinite) sum, can be expressed in terms of the kernel instead—this is the so-called *kernel trick*.

The RKHS $\mathcal{H}$ is just the space corresponding to multinomial expansions; the only novelty is that the expansion coefficients are basically the (reordered) Volterra coefficients in this framework.

The idea is simply to fit to training data $\{\underline{y}, X\}$, the best function of the form,

$$y_i = f(x_i, x_{i-1}, x_{i-2}, \dots, x_{i-M+1}), \tag{3.17}$$

where the memory $M$ is now treated as a hyperparameter. Then for the instant $i$, corresponding to time $t$, one has $\underline{z}(t) = \underline{z}_i = (x_i, x_{i-1}, x_{i-2}, \dots, x_{i-M+1})$ with $x_i = x(t)$, so that $y(t) = y_i = f(\underline{z})$.

As the problem has been framed, with $f \in \mathcal{H}$, one sees that

$$f(.) = \sum_{k=1}^{V_n} w_k \phi_k(.) \tag{3.18}$$

and,

$$y(t) = y_i = f(\underline{z}) = \sum_{k=1}^{V_n} w_k \phi_k(\underline{z}). \tag{3.19}$$

If the training data are $\{y_i, \underline{z}_i\} : i = 1, \dots, D$, the usual RKHS approximation gives,

$$y_i = f(\underline{z}_i) = \sum_{j=1}^{D} \alpha_j k(\underline{z}_i, \underline{z}_j). \tag{3.20}$$

To stress what has been gained, suppose one has specified a polynomial kernel

$$k(\underline{z}, \underline{z}') = (1 + < \underline{z}, \underline{z}' >)^L \tag{3.21}$$

or a squared-exponential kernel,

$$k(\underline{z}, \underline{z}') = \exp\left(-\frac{1}{l^2} < \underline{z}, \underline{z}' >\right). \tag{3.22}$$

Function evaluations for the direct expansion equation (3.8) involve a scalar product with $(L + M)!/L!M!$ multiplications and evaluation of the same number of basis functions. In contrast, for the expansion equation (3.20) with either of the kernels equations (3.21) or (3.20), one has $D$ multiplications and function evaluations, where each of the latter involves a scalar product with $M$

multiplications; the overall computational cost is $D(1+M)$ in terms of multiplications. As a concrete example, consider the case where one requires $M=30$ lags and a fifth-order nonlinearity, $L=5$. In this case, the number of multiplications in the direct expansion is $35!/30!5! = 324\,632$. Supposing that one has 1000 points of training data, the number of multiplications in the kernel expansion is 36 000—an order of magnitude smaller.[5] More interestingly perhaps, one should note that the expansion with the squared-exponential kernel—because it generates an expansion basis with polynomial terms of all orders—actually corresponds to an infinite *Volterra* expansion!

For a specific discrete Volterra modelling problem then, the kernel form is preferred on the grounds of computational expense and the problem becomes that of finding the coefficients $\alpha_i$ that give the best representation. Clearly, the best representation will require that one uses all the available training data, so that,

$$\hat{y}_i = \hat{f}(\underline{z}) = \sum_{i=1}^{D} \hat{\alpha}_i k(\underline{z}, \underline{z}_i), \tag{3.23}$$

where the carets indicate estimated/predicted quantities.

As is usual in problems of this type, the estimation will be framed as an optimization,

$$\hat{f}(\underline{z}) = \underset{f \in \mathcal{H}}{\operatorname{argmin}} \sum_{i=1}^{D} L\left(y_i - f(z_i)\right), \tag{3.24}$$

where $L$ is some convex loss function.

One mitigates against the effects of ill-conditioning or overtraining by adding a regularization term, so the problems becomes

$$\hat{f}(\underline{z}) = \underset{f \in \mathcal{H}}{\operatorname{argmin}} \left( \sum_{i=1}^{D} L\left(y_i - f(z_i)\right) + \frac{\rho}{2} \|f\|_{\mathcal{H}}^2 \right), \tag{3.25}$$

where $\rho \geq 0$ is a regularization (hyper) parameter.

In terms of the expansion equation (3.20), the minimization becomes,

$$\hat{\alpha} = \underset{\underline{\alpha} \in \mathbb{R}^D}{\operatorname{argmin}} \left( \sum_{i=1}^{D} L \left( y_i - \sum_{j=1}^{D} \alpha_j k\left(\underline{z}^i, \underline{z}\right) \right) + \frac{\rho}{2} \|f\|_{\mathcal{H}}^2 \right), \tag{3.26}$$

now in terms of $\alpha$. Of course, now the regularization term must also be expressed in terms of $\alpha$.

Using the properties of the RKHS, one sees that,

$$\begin{aligned} \frac{\rho}{2} \|f\|_{\mathcal{H}}^2 &= \frac{\rho}{2} \left\langle \sum_{i=1}^{D} \alpha_i k(\underline{z}^i, .), \sum_{j=1}^{D} \alpha_j k(\underline{z}^j, .) \right\rangle \\ &= \frac{\rho}{2} \sum_{i=1}^{D} \sum_{j=1}^{D} \alpha_i \alpha_j k(\underline{z}^i, \underline{z}^j) = \frac{\rho}{2} \underline{\alpha}^T K \underline{\alpha}, \end{aligned} \tag{3.27}$$

and the regularization term is rather neatly revealed to be a (weighted) weight-decay term [2], in the parameters $\alpha_i$.

The main optimization problem is thus,

$$\hat{\alpha} = \underset{\underline{\alpha} \in \mathbb{R}^D}{\operatorname{argmin}} \left( \sum_{i=1}^{D} L(y_i - \sum_{j=1}^{D} \alpha_j k(\underline{z}^i, \underline{z})) + \frac{\rho}{2} \underline{\alpha}^T K \underline{\alpha} \right), \tag{3.28}$$

and if one adopts the standard least-squares cost function, this becomes

$$\hat{\alpha} = \underset{\underline{\alpha} \in \mathbb{R}^D}{\operatorname{argmin}} \left( (\underline{y} - K\underline{\alpha})^T (\underline{y} - K\underline{\alpha}) + \frac{\rho}{2} \underline{\alpha}^T K \underline{\alpha} \right), \tag{3.29}$$

and this is minimized when $\partial J(\underline{\alpha})/\partial \underline{\alpha}^T = 0$. Expanding the cost function $J(\alpha)$ gives,

$$0 = \frac{\partial J}{\partial \alpha^T} = \frac{\partial}{\partial \alpha^T}\left(\underline{y}^T\underline{y} - \underline{\alpha}^T K^T \underline{y} - \underline{y}^T K \underline{\alpha} + \underline{\alpha}^T K^T K \underline{\alpha} + \frac{\rho}{2}\underline{\alpha}^T K \underline{\alpha}\right). \tag{3.30}$$

Now, treating the quantities $\underline{\alpha}^T$ and $\underline{\alpha}$ as independent in matrix calculus, and using the symmetry of $K$, one arrives at,

$$-K^T\underline{y} + K^T K \underline{\alpha} + \frac{\rho}{2} K^T \underline{\alpha} = 0, \tag{3.31}$$

which—using the symmetry of $K$ and assuming $K^T$ invertible—finally yields,

$$\left(K + \frac{\rho}{2}I\right)\underline{\alpha} = \underline{y}, \tag{3.32}$$

with solution,

$$\underline{\hat{\alpha}} = \left(K + \frac{\rho}{2}I\right)^{-1}\underline{y} \tag{3.33}$$

.

It is important to note at this point that the vector $\alpha$ does *not* quite contain the Volterra coefficients, and this was—after all—the point of the exercise; a little more work is needed.

One begins with the fact that the expansion

$$f(\underline{z}) = \sum_{k=1}^{V_n} w_k \phi_k(.\underline{z}) \tag{3.34}$$

*does* contain the Volterra coefficients; so, where is the inconsistency? The answer to that question requires a closer look at the expansion parameters. To see what is happening, it is enough to consider a quadratic expansion in two variables; these may represent $x_i$ and $x_{i-1}$, but will be denoted $x_1$ and $x_2$ here, for simplicity. To get a quadratic expansion in the two variables, the expansion basis need be $\underline{\phi}(\underline{x}) = (1, x_1, x_2, x_1x_2, x_1x_2, x_1^2, x_2^2)^{\mathrm{T}}$; however, any other basis where the terms are multiplied by scalars would suffice, and in fact, for consistency with the definition of the kernel, one does need to add scalars. Taking the polynomial kernel equation (3.21) with $L = 2$, one finds that,

$$k(\underline{z}, \underline{z}') = 1 + 2x_1x_1' + 2x_2x_2' + 2x_1x_1'x_2x_2' + (x_1x_1')^2 + (x_2x_2')^2, \tag{3.35}$$

and this needs to be consistent with equation (3.10), which can be conveniently rewritten as

$$k(\underline{z}, \underline{z}') = \sum_{k=1}^{V_n} \sqrt{\lambda_k}\phi_k(\underline{z})\sqrt{\lambda_k}\phi_k(\underline{z}'). \tag{3.36}$$

Direct comparison here shows that one needs to define the set of $\lambda_k$ as (1,2,2,2,1,1). The $\lambda_k$ are thus revealed as the multipliers needed to convert a multinomial basis into the form required for consistency with the kernel. It now becomes clear why $\alpha_i$ are not the same as the Volterra coefficients $w_i$; however, it is clear now how to convert from one to the other. One begins with equation (3.23) and substitutes from equation (3.10) to give

$$\hat{f}(\underline{z}) = \sum_{i=1}^{D} \alpha_i k(\underline{z}, \underline{z}_i) = \sum_{i=1}^{D} \alpha_i \sum_{k=1}^{V_n} \lambda_k \phi_k(\underline{z})\phi_k(\underline{z}^i). \tag{3.37}$$

Now, assuming one can interchange the order of the summations, one finds

$$\hat{f}(\underline{z}) = \sum_{k=1}^{V_n} \lambda_k \sum_{i=1}^{D} \alpha_i \phi_k(\underline{z})\phi_k(\underline{z}^i) = \sum_{k=1}^{V_n} w_k \phi_k(\underline{z}), \tag{3.38}$$

which identifies $w_k$ as

$$w_k = \lambda_k \sum_{i=1}^{D} \alpha_i \phi_k(\underline{z}^i). \tag{3.39}$$

In practice, $\lambda_i$ can also involve the hyperparameters of the kernel [15].

At this point all the necessary analysis is complete, and an illustration can be provided. Note that recent work in [24,25] has also arrived at an efficient means of casting Volterra series estimation as a regularized least-squares problem; however, their motivation was not from an RKHS viewpoint.

## (a) Numerical illustration

The algorithm used here was coded in MATLAB. The RKHS Volterra series coefficients were computed using equation (3.33); however, the hyperparameters $l$ and $\rho$ were optimized by minimizing the prediction error on the validation set. This optimization made use of the MATLAB function fmin, which uses a downhill-simplex algorithm. In this case, the model-predicted output (MPO) error was computed, where the predicted outputs are fed back; this is sometimes referred to as the simulation error in the electrical and control engineering communities, it is a more stringent test of validity than the one-step ahead (OSA) error. As an objective metric, the normalized mean-square error (NMSE) was used,

$$NMSE(y) = \frac{100}{D\sigma_y^2} \sum_{i=1}^{D} (y_i - \hat{y}_i)^2, \tag{3.40}$$

where $D$ is the number of training points and $\sigma_y^2$ is the variance of the measured displacements. This cost function has the following useful property; if the mean of the output signal is used as the model, i.e. $\hat{y}_i = \overline{y}$ for all $i$, the cost function is 100.0 (and can be thought of as a percentage). Any score less than 100% is thus evidence of captured correlation with the data; experience with this metric has shown that an NMSE less than 5% is evidence of a good model, while a score less than 1% shows an excellent result.

The data are simulated from a continuous-time Duffing oscillator as in equation (1.1), with $m = 1$, $c = 20$, $k = 10^4$ and $k_3 = 5 \times 10^9$. The excitation was chosen to be a zero-mean white Gaussian sequence with unit r.m.s. (band-limited onto the Nyquist interval) and the sampling interval $\Delta t$ was set at 0.008 s, corresponding to a sampling frequency of 125 Hz and a Nyquist frequency of 62.5 Hz.[6] As the undamped natural frequency of the underlying linear system is approximately 16 Hz, this sampling frequency is sufficient to capture the third harmonic in the response. At this level of excitation, the 'resonance' frequency in the naively estimated FRF of the system experienced a 6% shift from the underlying linear value and previous experience showed that this level of forcing excited the nonlinearity sufficiently for good identification results. Eight hundred points of data were simulated using a fourth-order Runge–Kutta scheme [26]. The first 200 points of data were discarded to eliminate transients and then the remaining 600 points were divided equally into training, validation and testing sets. Zero-mean Gaussian noise was added to the response data with s.d. of 10% of that of the data; this is a comparatively high level of noise. The input and output training data are shown in figure 1.

When the RKHS Volterra model was fitted to the data, the minimum validation error was achieved with $M = 19$ and $L = 3$. The other hyperparameters corresponding to these values were found to be $l = 1989.5$ and $\rho = 8.7687 \times 10^{-10}$.[7] A point of interest here is the 'high' number of lags; however, this is explainable. For a problem like Duffing's equation, which represents a single-degree-of-freedom oscillator, one would expect the number of lags to be related to the length of time for which the linear impulse response is non-zero. The reason for this is that the integral for $y_1(t)$ should approximately extend in time over the period for which $h_1(t)$ is non-zero. In this case, the true linear impulse response can be computed exactly and is shown in figure 2. The

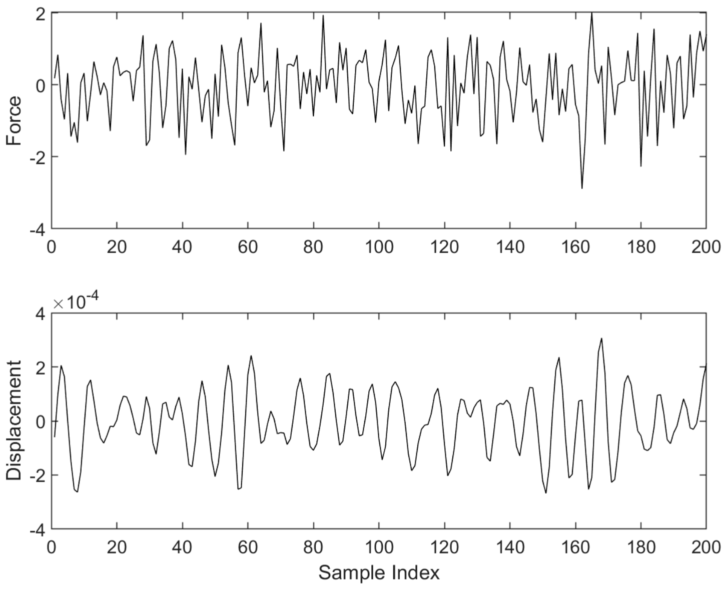

**Figure 1.** Input and output training data for case study.

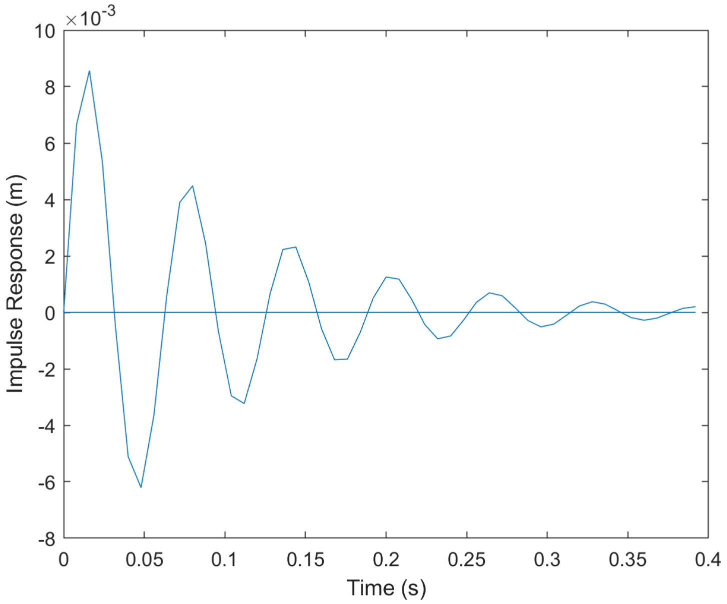

**Figure 2.** Linear impulse response function $h_1(t)$ for the Duffing oscillator of the case study.

figure shows that the impulse is non-zero up to 0.4 s, which corresponds to 50 lags; this means that $M = 19$ is actually quite low. The minimum of the validation error at $M = 19$ was initially thought to be because the Volterra series is truncated at third order here, so biased parameter estimates are produced; furthermore, the training data, and so on are contaminated by quite high noise.

Further investigation appeared to show some issues with the RKHS approach as implemented here. To investigate the extent of the bias induced by the truncated series, the first-order kernel (impulse response) was computed using equation (3.39), and the results deviated quite significantly from the expected result given in figure 2. The results also proved quite sensitive to the model hyperparameters. Thus, the coefficients of the RKHS are not in good correspondence with

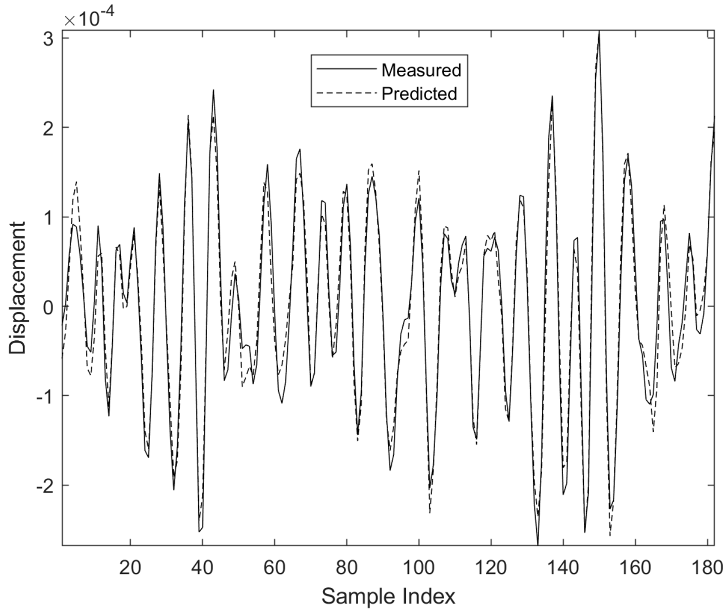

**Figure 3.** Comparison between measured and predicted responses for RKHS Volterra model for case study: training data.

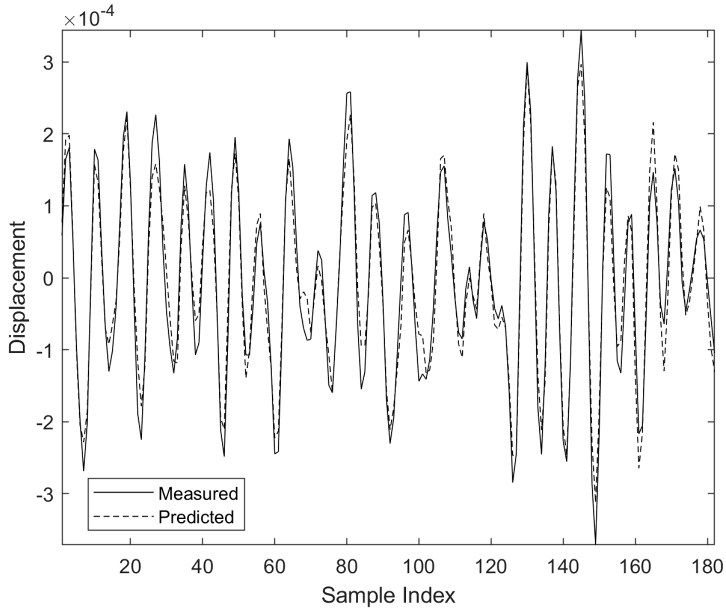

**Figure 4.** Comparison between measured and predicted responses for RKHS Volterra model for case study: validation data.

the actual kernels; however, the model predictions in the time domain are very respectable. Figures 3–5 show comparisons between the measured and predicted responses for the training, validation and test sets, respectively; the corresponding NMSE values are 4.0994, 5.1359 and 5.3312. These values are not excellent, but the result is to expected given the extent of the noise.

Finally, it must be recognized that the models are very highly parameterized and overtraining is an issue. The conclusion here is that the RKHS method can give good predictive models, but these models should probably be regarded as ’black-box’ learners and the outcome can be system modelling rather than SI. While it is certainly possible to achieve ’physical results’ as evidenced by HFRF estimates as in [15], it is also true to say that one can turn to other machine-learning

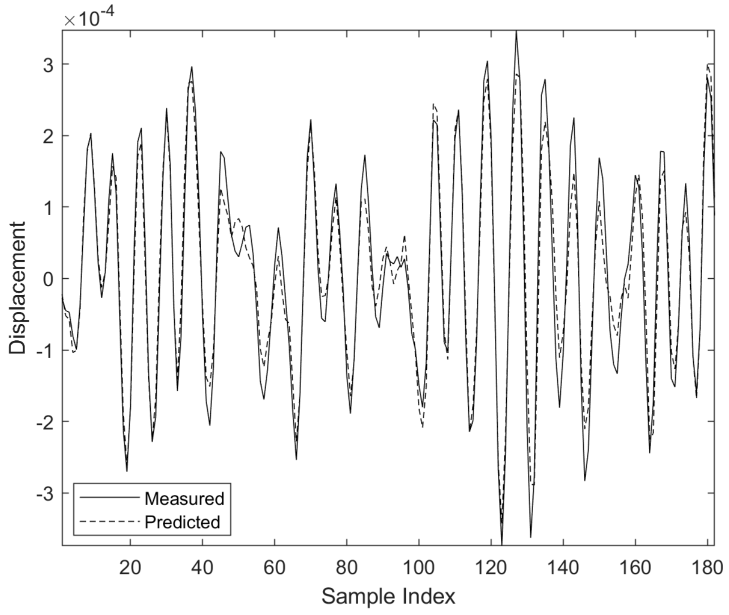

**Figure 5.** Comparison between measured and predicted responses for RKHS Volterra model for case study: test data.

paradigms to get arguably more robust methods. In the next section, the Volterra series will appear in the context of a new machine-learning model class which allows much more parsimonious and accurate models.

## 4. Gaussian process NARX models

### (a) Gaussian processes

The basic premise of GPs is to perform inference over functions directly, as opposed to inference over parameters of functions. In short, a GP is a distribution over functions, which is conditioned on training data so that the most probable functions are the best fits to the data.

Let $X = [\underline{x}_1, \underline{x}_2 \dots \underline{x}_D]^T$ denote a matrix of multivariate training inputs, and $\underline{y}$ denote the corresponding vector of training outputs. The input vector for a testing point will be denoted by the column vector $\underline{x}^*$ and the corresponding (unknown) output by $y^*$. A GP prior is formed by assuming a (Gaussian) distribution over functions

$$f(\underline{x}) \sim \mathcal{GP}\left(m(\underline{x}), k(\underline{x},\underline{x})\right), \tag{4.1}$$

where $m(\underline{x})$ is the mean function and $k(\underline{x},\underline{x}')$ is a positive-definite covariance function. As a regression model, the GP fits a relationship $y = f(\underline{x}) + \epsilon$, where $\epsilon$ is a noise process drawn from a univariate zero-mean Gaussian distribution $N(0, \sigma_n^2)$.

One of the defining properties of the GP is that the density of a finite number of outputs from the process, both observed and unobserved, is multivariate normal. This property, combined with standard results for Gaussian distributions, can be used to condition unobserved points on observed training points: this mechanism effectively fits the GP to the training data.

Following a Bayesian approach, the prior mean is assumed to be zero (see [27] for a discussion). Assuming a Gaussian noise model with variance $\sigma_n^2$, the joint distribution for training and testing values is

$$\begin{pmatrix} \underline{y} \\ y^* \end{pmatrix} \sim \mathcal{N}\left(\underline{0}, \begin{bmatrix} K(X,X) + \sigma_n^2 & K\left(X, \underline{x}^*\right) \\ K\left(\underline{x}^*, X\right) & K\left(\underline{x}^*, \underline{x}^*\right) + \sigma_n^2 \end{bmatrix}\right), \tag{4.2}$$

where $K(X,X)$ is a matrix whose $i,j$th element is equal to $k(\underline{x}_i, \underline{x}_j)$. Similarly, $K(X, \underline{x}^*)$ is a column vector whose $i$th element is equal to $k(\underline{x}_i, \underline{x}^*)$, and $K(\underline{x}^*, X)$ is the transpose of the same; $\mathbb{I}$ is the identity matrix.

To make use of the above, it is necessary to rearrange the joint distribution $p(\underline{y}, y^*)$ into a conditional distribution $p(y^* | \underline{y})$. Using standard results for the conditional properties of a Gaussian reveals [27]

$$y^* \sim \mathcal{N}(m^*(\underline{x}^*), k^*(\underline{x}^*, \underline{x}^*)), \tag{4.3}$$

where

$$m^*(\underline{x}^*) = K(\underline{x}^*, X)[k(X,X) + \sigma_n^2 \mathbb{I}]^{-1} \underline{y} \tag{4.4}$$

is the *posterior mean* of the GP and

$$k^*(\underline{x}^*, \underline{x}') = k(\underline{x}^*, \underline{x}') - K(\underline{x}^*, X)[K(X,X) + \sigma_n^2 \mathbb{I}]^{-1} K(X, \underline{x}') \tag{4.5}$$

is the posterior variance.

Thus the GP model provides a full posterior distribution for the unknown quantity $y^*$. The posterior mean from equation (4.3) can then be used as a 'best estimate' for a regression problem, and the posterior variance can also be used to define confidence intervals. The covariance function used here is the squared-exponential function:

$$k(\underline{x}, \underline{x}') = \sigma_f^2 \exp\left(-\frac{||\underline{x} - \underline{x}'||^2}{2l^2}\right). \tag{4.6}$$

The problem now has three *hyperparameters*: $l$ is a characteristic length scale, the multiplier $\sigma_f^2$ gives overall scale and is sometimes called the *height parameter*, $\sigma_n^2$ estimates the noise variance and the term acts as a regulariser. For considerably more details on GPs than this paper allows, see [27].

## (b) GP-NARX models

The GP model above is a *static* map, learning the relationship between point inputs and point outputs. However, it is almost trivial to learn dynamical system behaviour, simply by adopting a NARX framework. Over the last 30 years, one of the most versatile and enduring time-series models used for NLSI has been the nonlinear auto-regressive moving average with exogenous inputs (NARMAX) model. The NARMAX model was introduced in 1985 [28,29] and has been the subject of constant interest and development since. (A comprehensive monograph on the theory and applications of the model fairly recently appeared in [30].) The most general model form accommodates nonlinear discrete-time process and noise models. However, if the noise process can be assumed to be white Gaussian, the simpler NARX model can be adopted, and this will be the focus of this section. The NARX model assumes a form whereby the current value of the system output is predicted using a nonlinear function $F$ of previous inputs and outputs, i.e.

$$y_i = F(y_{i-1}, \dots, y_{i-n_y}; x_i, \dots, x_{i-n_x+1}) + \epsilon_i, \tag{4.7}$$

where the *residual sequence* $\epsilon_i$ is white Gaussian. The number of output (respectively, input) lags is denoted $n_y$ (respectively, $n_x$). In the analysis here, the functional form in equation (4.7) adopts a function $F$ learnt from data using a GP. Learning the function is straightforward, the only subtlety required is that the lagged inputs and outputs need to be assembled into vectors and matrices suitable for the application of the GP algorithm as described earlier.

Because the modelling algorithm is being used in 'anger' here, it will be important to assess the validity of the model in a form appropriate to the assessment of time-series predictions. There are various tests one can apply to assess the validity of a time-series model; the most basic option is to compute OSA predictions. In this case, using the training data, one computes the predictions for a given time using observed inputs and outputs up to that time, i.e.

$$y_i^* = F(y_{i-1}, \dots, y_{i-n_y}; x_i, \dots, x_{i-n_x+1}), \tag{4.8}$$

and compares the predicted and observed outputs.

It is always useful to have an objective measure of comparison, and the one used here will be the NMSE defined by equation (3.40).

Clearly, the OSA predictions are not a particularly stringent test of the model. A more demanding test is to compute the MPO defined by,

$$y_i^* = F(y_{i-1}^*, \dots, y_{i-n_y}^*; x_i, \dots, x_{i-n_x+1}), \tag{4.9}$$

and this test can be conducted on testing data as well as training data, which is an important consideration in the more general context of machine learning. Various correlation functions also provide a stringent means of validating models [11]; however, they are not employed here.

The GP form for the NARX model has its advantages and disadvantages; two of the main issues are discussed briefly here, with directions to the literature as to their possible means of solution. The first problem is—as mentioned previously—that the GP algorithm depends on the inversion of the covariance matrix $K$; this is an operation which costs $O(D^3)$ multiplications, where $D$ is the number of training points.[8] In fact, SI with NARX models has traditionally been carried out with small training sets with a low number of thousands of data points, and this size of problem is typically feasible using a standard GP algorithm. However, if one wishes to move to larger training sets, the costs of computation can become prohibitive. This problem has led various ideas on reducing the burden, good references are [31,32]. The second problem with the GP-NARX formulation relates to noise on the training data. The standard formulation assumes that the training inputs are noise-free and that the noise on the outputs is Gaussian with constant variance as discussed above. This can be an issue if one is attempting multi-step ahead predictions with a GP-NARX model; because of the feeding back of the output predictions, the outputs *become* inputs and carry their predictive uncertainty with them. One of the first comprehensive studies of this problem appears to have been the work leading to the thesis [33]. Closed-form approximate solutions for the predictive mean and variance in the presence of input noise can be found in [34]. In the case study presented here (for reasons discussed later), the issues referred to above have been ignored without (it is believed) damage to the results; however, in other engineering problems, they will probably need to be addressed.

### (i) Case study—an asymmetric duffing oscillator

To illustrate the use of the GP-NARX formulation, data simulated from a Duffing oscillator data system are used. In the asymmetric case when a quadratic stiffness is present, the relevant equation of motion is

$$m\ddot{y} + c\dot{y} + ky + k_2y^2 + k_3y^3 = x(t). \tag{4.10}$$

Data were simulated here by integrating the equation of motion using a fourth-order fixed-step Runge–Kutta algorithm [26]. The parameters adopted were $m = 1$, $c = 20$, $k = 10^4$, $k_2 = 10^7$ and $k_3 = 5 \times 10^9$. The excitation used was a zero-mean white Gaussian random sequence with a s.d. of 2.0, band-limited on to the Nyquist interval. The time step used was $\Delta t = 0.001$ s corresponding to a sampling frequency of 1 kHz. As before, this level of forcing sufficiently excited the nonlinearity. Gaussian noise of 1% r.m.s. of the signal was added to the response time data. The results presented here are for an independent test set of data, also comprising 1000 samples of data from the system at the same level of excitation as the training data and with the same amplitude of added noise. The three main hyperparameters for the simple GP formulation with a squared exponentional kernel used here were determined by using a conjugate-gradients algorithm to maximize the log marginal evidence [27]. It was also necessary to establish the number of input and output

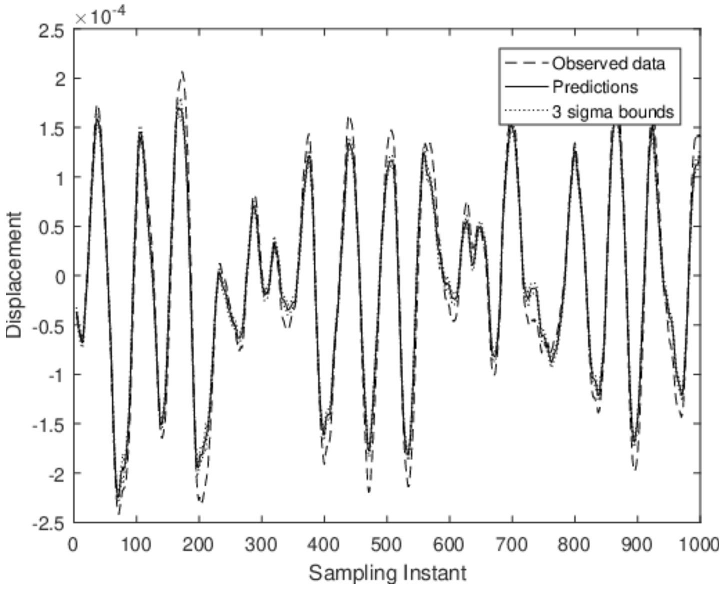

**Figure 6.** MPO predictions for GP-NARX model of Duffing oscillator data.

lags needed in the model; these numbers are also hyperparameters of the GP-NARX model. A quick search using the errors on a validation set gave the values $n_x = n_y = 3$. Once the lag numbers were established, the GP-NARX model was fitted and the optimal GP hyperparameters were found to be: $\sigma_f^2 = 757.4$, $l = 9.581$ and $\sigma_n^2 = 3.057 \times 10^{-4}$. To improve the conditioning of the estimation process, all data were standardized before the computation, the scales for the data were reintroduced after predictions were made.

As discussed above, the MPO predictions provide the most stringent test, and these are shown in figure 6. The corresponding NMSE in this case was 3.44, which still indicates a good fit.

Note that the confidence intervals are very small and do not accommodate the observed prediction errors; this is because not all of the uncertainty has been accounted for. In the predictions so far, the predicted outputs have been fed back into the model to form the MPO. This means that the only uncertainty accounted for in the predictions is the parameter uncertainty. To take a proper Bayesian viewpoint, one should allow for the fact that each prediction is actually a sample from a distribution; this distribution being determined by the parameter distribution. To account for this, during a prediction run, at each instant $i$, the prediction $y_i^*$ was sampled from the distribution specified by the predictive mean and covariance as specified by equations (4.4) and (4.5). One such run generates a single realization of the prediction process, to accumulate information about the distribution of predictions with state estimation taken into account, a Monte Carlo approach was adopted here with 25 different runs conducted. Figure 7 shows the 25 realizations of the predictions.

There is clearly a great deal more uncertainty associated with the predictions now. From the Monte Carlo realizations, one can estimate a mean prediction and determine $\pm 3\sigma$ confidence bounds, and the result of the analysis for the case here is shown in figure 8. The confidence intervals are now a more appropriate assessment of the predictive capability of the model. This exercise shows clearly that the dominant contribution to uncertainty in the predictions is not the direct component from the parameter uncertainty, but the indirect component because of state estimation from the uncertain parameters.

Having established a benchmark dataset and illustrated the GP-NARX performance, it is possible to show how GP-NARX models can be extended in their use to provide a powerful means of estimating Volterra HFRSs and thus visualizing nonlinear response.

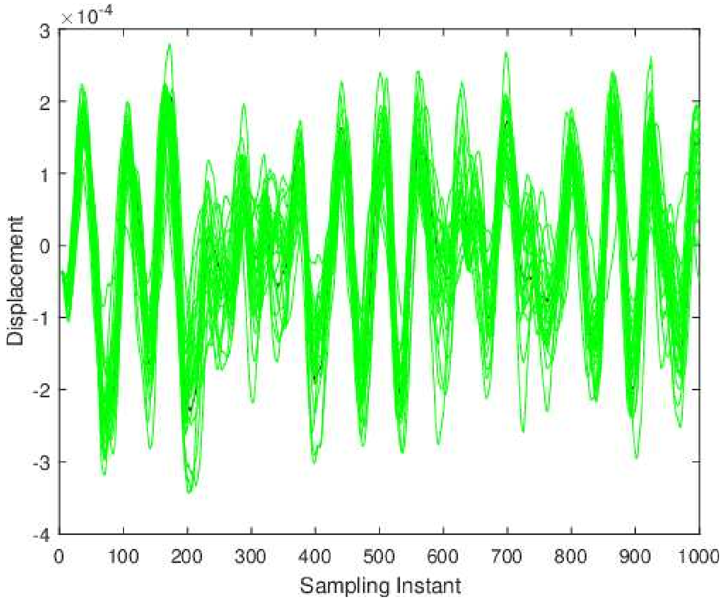

**Figure 7.** Monte Carlo realizations of predictions for GP-NARX model of Duffing oscillator data. The black line is the measured output.

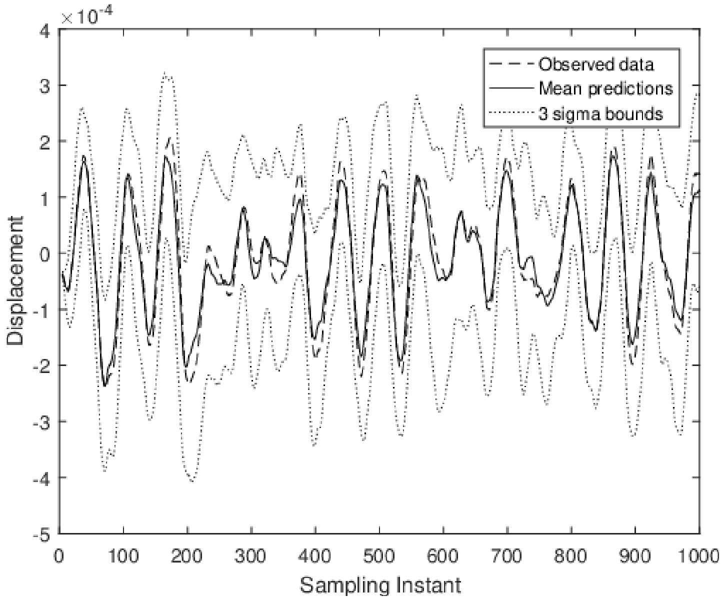

**Figure 8.** Monte Carlo predictions for GP-NARX model of Duffing oscillator data.

## (c) Higher-order FRFs of the GP-NARX model

Given a time-series model of a system, one can extract the corresponding Volterra FRFs by a process called *Harmonic Probing*. This algorithm was introduced in [35] for continuous-time systems and extended to discrete-time systems in [11]; only the briefest introduction is given here; more details and worked examples can be found in [1]. The basic idea is quite simple; for a linear system, if one 'probes' the equation of motion with a harmonic input $e^{i\omega t}$, the response can be shown to be $H_1(\omega)e^{i\omega t}$. Extraction of $H_1(\omega)$ is a matter of straightforward algebra. The same principle extends to nonlinear systems; for example, if the probing input $e^{i\omega_1 t} + e^{i\omega_2 t}$ is used, the response can be shown to be $H_1(\omega_1)e^{i\omega_1 t} + H_1(\omega_1)e^{i\omega_2 t} + 2H_2(\omega_1,\omega_2)e^{i(\omega_1+\omega_2)t}$ + higher-order terms; the algebra is a little more complicated, but $H_2$ can be extracted. The same principle can be applied for all HFRFs, although the analysis gets more demanding.

Before proceeding to the GP-NARX model, it is necessary to determine the explicit form of the GP-NARX model on an order-by-order basis. First of all, one observes that the GP can be expressed as an expansion in terms of basis functions fixed by the covariance kernel and the training data [17], the predicted output $y^*$ corresponding to a new input $\underline{x}^*$ is given by

$$y^* = \sum_{i=1}^{N} a_i k(\underline{x}^*, \underline{x}_i), \tag{4.11}$$

where,

$$\underline{a} = [k(X,X) + \sigma_n^2 I]^{-1}\underline{y}, \tag{4.12}$$

and this is fixed by the training data.[9] If one adopts the squared exponential covariance function, one arrives at the GP-NARX form,

$$y_i = \sigma_f^2 \sum_{j=1}^{N} a_j \exp\left\{-\frac{1}{2l^2}\left[\sum_{k=1}^{n_y}(y_{i-k} - v_{jk})^2 + \sum_{m=0}^{n_x}(x_{i-m} - u_{jm})^2\right]\right\}, \tag{4.13}$$

where the matrix $V = \{v_{ij}\}$ is formed from the first $n_y$ columns of the matrix $X$ and $U = \{u_{ij}\}$ is formed from the remaining $n_x + 1$ columns of $X$. Note that the notation adopted means that an input *vector* $\underline{x}$ will have components which are lagged system inputs and outputs.

Note too that this expression is essentially that of the radial-basis function neural network considered in [13]; this means that the HFRFs derived in that paper are applicable here. However, the analysis here presents a more direct approach in terms of homogeneous autoregressive exogenous (ARX) and NARX model coefficients at each polynomial order; the expressions here also correct some typographical errors in [13].

The first issue which arises is that the function in equation (4.13) must be expanded as a polynomial to apply harmonic probing. As observed in [13], direct expansion means that the term of order $n$ will contain powers of all orders up to $n$ and this makes it impossible to group linear terms etc. The solution is simple, a trivial rearrangement yields the more amenable form:

$$y_i = \sigma_f^2 \sum_{j=1}^{N-p} a_j \gamma_j \exp\left\{-\frac{1}{2l^2}\left[\sum_{k=1}^{n_y}(y_{i-k}^2 - 2v_{jk}y_{i-k}) + \sum_{m=0}^{n_x}(x_{i-m}^2 - 2u_{jm}x_{i-m})\right]\right\}, \tag{4.14}$$

where

$$\gamma_j = \exp\left\{-\frac{1}{2l^2}\left[\sum_{k=1}^{n_y} v_{jk}^2 + \sum_{m=0}^{n_x} u_{jm}^2\right]\right\}. \tag{4.15}$$

As discussed above, to identify $H_1(\omega)$, the system is 'probed' with the expression

$$x_i^p = e^{i\Omega t} \tag{4.16}$$

and this yields a response

$$y_i^p = H_1(\Omega)e^{i\Omega t} + H_2(\Omega,\Omega)e^{2i\Omega t} + H_3(\Omega,\Omega,\Omega)e^{3i\Omega t} + \cdots. \tag{4.17}$$

If the coefficient of $e^{i\Omega t}$ is extracted from the expression, for $y_i^p$, the only HFRF which can appear is $H_1(\Omega)$; thus the expression can be rearranged to give an analytical expression for $H_1$. In fact, one need only consider the linear terms in the expansion to extract $H_1$, so one essentially considers the ARX model,

$$y_i = \sigma_f^2 \sum_{j=1}^{D-p} \frac{a_j \gamma_j}{l^2} \left\{ \sum_{k=1}^{n_y} v_{jk} y_{i-k} + \sum_{m=0}^{n_x} u_{jm} x_{i-m} \right\}. \tag{4.18}$$

Changing the order of summation here results in the standard ARX form:

$$y_i = \sum_{j=1}^{n_y} \alpha_j y_{i-j} + \sum_{j=0}^{n_x} \beta_j x_{i-j}, \tag{4.19}$$

where,

$$\alpha_j = \frac{\sigma_f^2}{l^2} \sum_{i=1}^{D-p} a_i \gamma_i v_{ij}, \tag{4.20}$$

$$\beta_j = \frac{\sigma_f^2}{l^2} \sum_{i=1}^{D-p} a_i \gamma_i u_{ij}. \tag{4.21}$$

Harmonic probing of this expression is straightforward; one substitutes the probing expressions equation (4.16) and (4.17) into equation (4.19) and collects together all the coefficients of $e^{i\Omega t}$. In doing this, account must be taken of the effect of time delays on the harmonic signals, this is straightforward to compute as

$$x_{i-k} = B^k x_i = B^k e^{i\Omega t} = e^{-ki\Omega \Delta t} e^{i\Omega t}, \tag{4.22}$$

$$y_{i-k} = B^k y_i = B^k H_1(\Omega) e^{i\Omega t} = e^{-ki\Omega \Delta t} H_1(\Omega) e^{i\Omega t}, \tag{4.23}$$

where $B$ is the backward shift operator. The result of the calculation is,

$$H_1(\Omega) = \frac{\sum_{j=0}^{n_x} \beta_j e^{-ij\Delta t \Omega}}{1 - \sum_{j=1}^{n_y} \alpha_j e^{-ij\Delta t \Omega}}, \tag{4.24}$$

with $\alpha_j$ and $\beta_j$ as defined in equations (4.20) and (4.21).

The extraction of $H_2$ is a little more complicated; as mentioned above, this requires probing with two independent harmonics, so one applies $x_i^p = e^{i\Omega_1 t} + e^{i\Omega_2 t}$, which results in $y_i^p = H_1(\Omega_1) e^{i\Omega_1 t} + H_1(\Omega_2) e^{i\Omega_2 t} + 2H_2(\Omega_1, \Omega_2) e^{i(\Omega_1 + \Omega_2)t} + \ldots$. If these expressions are substituted into the GP function equation (4.14), the only HFRFs to appear in the coefficient of the sum harmonic $e^{i(\Omega_1+\Omega_2)t}$, are $H_1$ and $H_2$, where $H_1$ is already known from equation (4.24). As before, the coefficient can be rearranged to give an expression for $H_2$ in terms of the GP parameters and $H_1$. The only terms in the expansion of equation (4.14) which are relevant for the calculation are those at first and second order. The calculation is straightforward but tedious and yields,

$$H_2(\Omega_1, \Omega_2) = \frac{A + B + C}{E}, \tag{4.25}$$

where,

$$A = \sum_{k=1}^{n_y} \sum_{l=1}^{n_y} \alpha_{kl} H_1(\Omega_1) H_1(\Omega_2) \left( e^{-i\Omega_1 k \Delta t} . e^{-i\Omega_2 l \Delta t} + e^{-i\Omega_2 k \Delta t} . e^{-i\Omega_1 l \Delta t} \right), \tag{4.26}$$

$$B = \sum_{k=1}^{n_y} \sum_{l=0}^{n_x} \beta_{kl} \left( H_1(\Omega_1) e^{-i\Omega_1 k \Delta t} . e^{-i\Omega_2 l \Delta t} + H_1(\Omega_2) e^{-i\Omega_2 k \Delta t} . e^{-i\Omega_1 l \Delta t} \right), \tag{4.27}$$

$$C = \sum_{k=0}^{n_x} \sum_{l=0}^{n_x} \gamma_{kl} \left( e^{-i\Omega_1 k\Delta t}.e^{-i\Omega_2 l\Delta t} + e^{-i\Omega_2 k\Delta t}.e^{-i\Omega_1 l\Delta t} \right) \tag{4.28}$$

and

$$E = 1 - \sum_{k=1}^{n_y} \alpha_k e^{-i(\Omega_1+\Omega_2)k\Delta t}. \tag{4.29}$$

The coefficients in the above expressions are given by

$$\alpha_{jm} = \frac{\sigma_f^2}{4l^4} \sum_{i=1}^{D-p} a_i \gamma_i v_{ij} v_{im} - \delta_{jm} \frac{\sigma_f^2}{2l^2} \sum_{i=1}^{D-p} a_i \gamma_i, \tag{4.30}$$

$$\beta_{jm} = \frac{\sigma_f^2}{2l^4} \sum_{i=1}^{D-p} a_i \gamma_i v_{ij} u_{im}, \tag{4.31}$$

$$\gamma_{jm} = \frac{\sigma_f^2}{4l^4} \sum_{i=1}^{D-p} a_i \gamma_i u_{ij} u_{im} - \delta_{jm} \frac{\sigma_f^2}{2l^2} \sum_{i=1}^{D-p} a_i \gamma_i, \tag{4.32}$$

where $\delta_{jm}$ is the standard Kronecker delta.

Derivation of $H_3$ is considerably more lengthy and requires probing with three harmonics, the expression is not given here for reasons of space. The results here will only present examples of these calculations for $H_1$ and $H_2$.

## (d) HFRF results for the duffing oscillator case study system

The HFRFs for the asymmetric Duffing oscillator system of equation (4.10) are estimated from a GP-NARX model fitted to the simulated data. As the objective here is to compare the HFRF estimates with exact forms derived from Duffing's equation, a dataset was analysed where only 0.001% noise was added to the Duffing response data.

As before, the GP hyperparameters were estimated by maximizing the log marginal evidence, in this case the results were: $\sigma_f^2 = 129.2$, $l = 8.027$ and $\sigma_n^2 = 5.54 \times 10^{-11}$. The model gave an OSA error of $9.4 \times 10^{-7}$ and an MPO error of 0.001. The comparisons between predicted and measured response are not given as the curves are not distinguishable given the accuracy of the predictions. However, it is meaningful to give comparisons between the exact HFRFs—which can be found in [1]—and those estimated from the GP. Figure 9 shows a comparison between the exact and estimated $H_1(\omega)$; it is clear that the estimate is very accurate indeed.

Figures 10 and 11 show comparisons between the exact and estimated $H_2$ functions in terms of magnitude and phase, respectively. Because a direct visual comparison is subjective when the surfaces are displayed, the exact and estimated diagonals $H_2(\omega, \omega)$ are shown in figure 12, the accuracy of the estimates is clearly excellent.

The work presented so far has shown that models based on machine learning allow direct estimation of Volterra kernels and HFRFs with only moderate computational costs. However, all the results are for single-input single-output systems $x(t) \longrightarrow y(t)$. In the final section we present new results showing that the analysis is extendable to MIMO systems by using an extended harmonic probing algorithm [36].

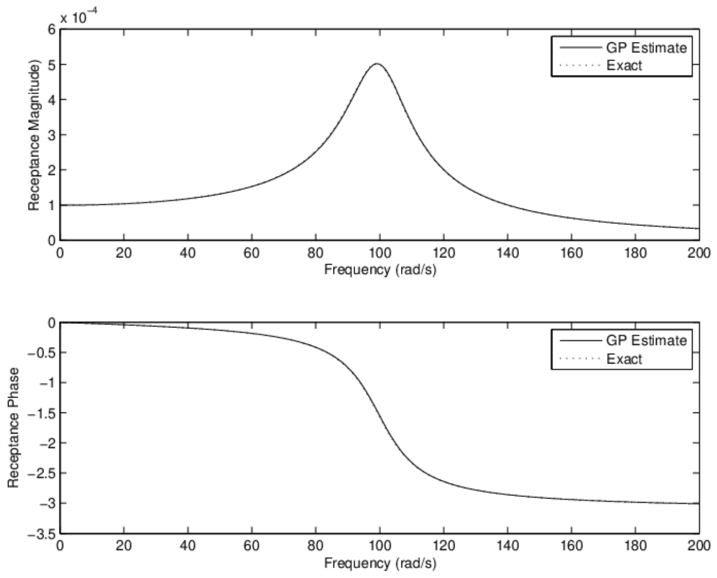

**Figure 9.** GP estimate of $H_1(\omega)$ compared to exact result.

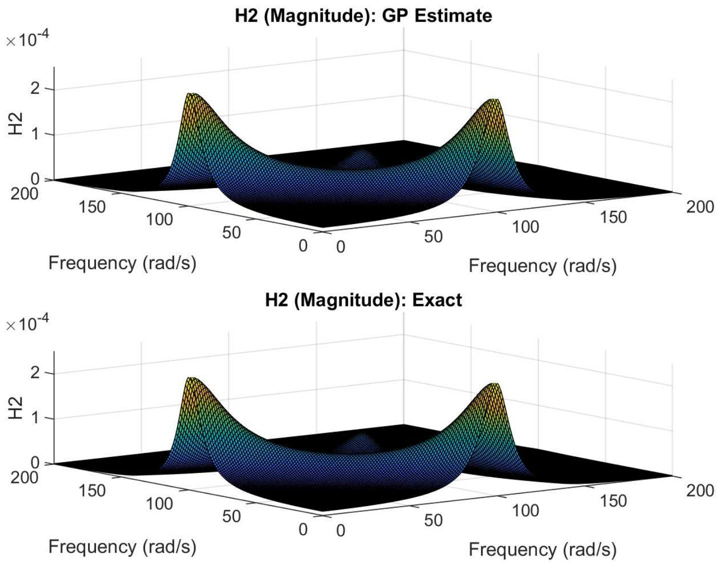

**Figure 10.** GP estimate of $H_2(\omega_1, \omega_2)$ magnitude compared to exact result.

## 5. HFRFs from Multi-Input Multi-Output (MIMO) systems

### (a) The MIMO Volterra series

This section will consider the general case where a system may be stimulated at $M$ points and can respond at $N$ points; in fact, although the algebra can become considerably more complicated, a great deal of the complexity is a matter of bookkeeping. The extension to the harmonic probing algorithm is fairly straightforward.

For the purposes of establishing notation, it is simplest to begin with a MIMO linear system. In the linear case, with inputs $\{x^{(i)} : i = 1, \dots, M\}$ and outputs $\{y^{(i)} : i = 1, \dots, N\}$, the principle of

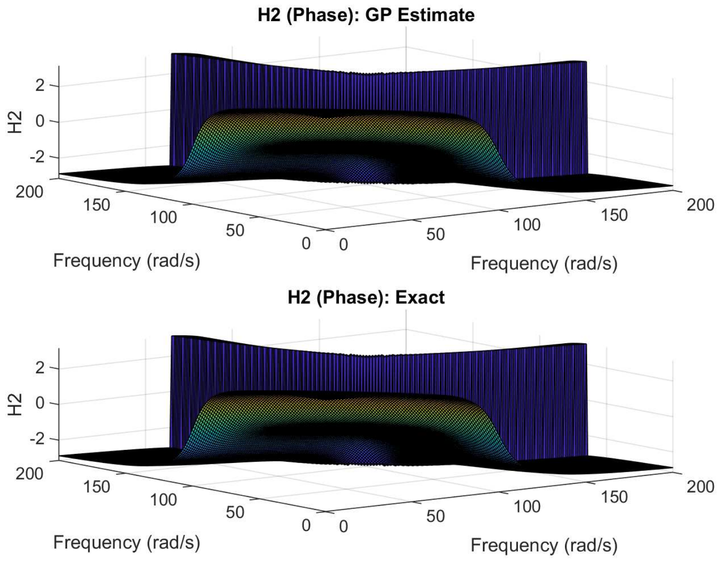

**Figure 11.** GP estimate of $H_2(\omega_1, \omega_2)$ phase compared to exact result.

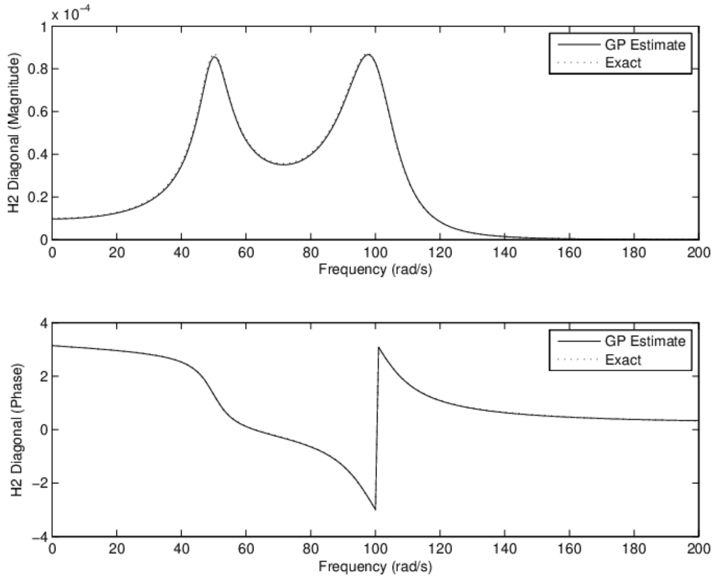

**Figure 12.** GP estimate of $H_2(\omega, \omega)$ magnitude and phase compared to exact result.

superposition [1], shows that any given response can be expanded as,

$$y_1^{(p)}(t) = \int_0^\infty h_1^{(p:1)}(\tau)x^{(1)}(t-\tau)d\tau + \cdots + \int_0^\infty h_1^{(p:M)}(\tau)x^{(n)}(t-\tau)d\tau, \tag{5.1}$$

and the complete system is characterized by an $N \times M$ matrix of impulse responses $h_1^{(i:j)}$, or alternatively, an $N \times M$ FRF matrix. In the general nonlinear case, the terms $y_1^{(p)}$ will be just the first terms in a set of infinite series $y^{(p)} = y_1^{(p)} + \ldots y_n^{(p)} + \ldots$. The responses at a specific point will be composed of a sum of contributions over different inputs and different nonlinear orders. To get an impression of how this works, it is sufficient to consider the second-order contribution at a given point; for further simplicity, the two-input case will be considered. In this case, one has

$$\begin{aligned}y_2^{(p)}(t) = &\int_0^\infty \int_0^\infty h_2^{(p:aa)}(\tau_1,\tau_2)x^{(a)}(t-\tau_1)x^{(a)}(t-\tau_2)d\tau_1 d\tau_2\\ &+\int_0^\infty \int_0^\infty h_2^{(p:ab)}(\tau_1,\tau_2)x^{(a)}(t-\tau_1)x^{(b)}(t-\tau_2)d\tau_1 d\tau_2\\ &+\int_0^\infty \int_0^\infty h_2^{(p:ba)}(\tau_1,\tau_2)x^{(b)}(t-\tau_1)x^{(a)}(t-\tau_2)d\tau_1 d\tau_2\\ &+\int_0^\infty \int_0^\infty h_2^{(p:bb)}(\tau_1,\tau_2)x^{(b)}(t-\tau_1)x^{(b)}(t-\tau_2)d\tau_1 d\tau_2.\end{aligned} \tag{5.2}$$

For notational simplicity, groups of kernels referring to the same input combinations can be combined; in this case, one can take,

$$h_2^{(p:ab)}(\tau_1,\tau_2)+h_2^{(p:ba)}(\tau_2,\tau_1)\longrightarrow 2h_2^{(p:ab)}(\tau_1,\tau_2), \tag{5.3}$$

which simplifies equation (5.2) to

$$\begin{aligned}y_2^{(p)}(t) = &\int_0^\infty \int_0^\infty h_2^{(p:aa)}(\tau_1,\tau_2)x^{(a)}(t-\tau_1)x^{(a)}(t-\tau_2)d\tau_1 d\tau_2\\ &+2\int_0^\infty \int_0^\infty h_2^{(p:ab)}(\tau_1,\tau_2)x^{(a)}(t-\tau_1)x^{(b)}(t-\tau_2)d\tau_1 d\tau_2\\ &+\int_0^\infty \int_0^\infty h_2^{(p:bb)}(\tau_1,\tau_2)x^{(b)}(t-\tau_1)x^{(b)}(t-\tau_2)d\tau_1 d\tau_2.\end{aligned} \tag{5.4}$$

The combinations become a little more interesting at higher order; for example, in the $h_3$ case with two different inputs, one has,

$$3h_3^{(p:aab)}(\tau_1,\tau_2,\tau_3)=h_3^{(p:aab)}(\tau_1,\tau_2,\tau_3)+h_3^{(p:aba)}(\tau_1,\tau_3,\tau_2)+h_3^{(p:baa)}(\tau_3,\tau_2,\tau_1), \tag{5.5}$$

whereas when a unique excitation is applied at three different points, one has,

$$\begin{aligned}6h_3^{(p:abc)}(\tau_1,\tau_2,\tau_3)=&h_3^{(p:abc)}(\tau_1,\tau_2,\tau_3)+h_3^{(p:acb)}(\tau_1,\tau_3,\tau_2)+h_3^{(p:bac)}(\tau_2,\tau_1,\tau_3)\\ &+h_3^{(p:bca)}(\tau_2,\tau_3,\tau_1)+h_3^{(p:cab)}(\tau_3,\tau_1,\tau_2)+h_3^{(p:cba)}(\tau_3,\tau_2,\tau_1).\end{aligned} \tag{5.6}$$

Now, under the usual multidimensional Fourier transformation, each of these Volterra kernels will give a distinct FRF; for example, corresponding to $h_3^{(p:abc)}(\tau_1,\tau_2,\tau_3)$ there will be a $H_3^{(p:abc)}(\omega_1,\omega_2,\omega_3)$. In this much more general case, harmonic probing works exactly as before, except that several different probing expressions are needed at each nonlinear order to single out specific kernels. The first-order case is simple, the required expressions are,

$$x^{(a)}(t)=e^{i\Omega_1 t}+e^{i\Omega_2 t}+e^{i\Omega_3 t}, \tag{5.7}$$

$$\begin{aligned}y^{(p)}(t)=&H_1^{(p:a)}(\Omega_1)e^{i\Omega_1 t}+H_1^{(p:a)}(\Omega_2)e^{i\Omega_2 t}+H_1^{(p:a)}(\Omega_3)e^{i\Omega_3 t}+2H_2^{(p:aa)}(\Omega_1,\Omega_2)e^{i(\Omega_1+\Omega_2)t}\\ &+2H_2^{(p:aa)}(\Omega_1,\Omega_3)e^{i(\Omega_1+\Omega_3)t}+2H_2^{(p:aa)}(\Omega_2,\Omega_3)e^{i(\Omega_2+\Omega_3)t}\\ &+6H_3^{(p:aaa)}(\Omega_1,\Omega_2,\Omega_3)e^{i(\Omega_1+\Omega_2+\Omega_3)t}+\cdots,\end{aligned} \tag{5.8}$$

and this is sufficient to extract the first-order HRFs as one progresses over the input points $a$. However, it is important to note that probing at a single input point can only generate 'diagonal'

$H_2$ combinations; i.e., $H_2^{(p:aa)}$. To extract a complete set of $H_2$ HFRFs, one needs to simultaneously probe at more than one input point; for example,

$$x^{(a)}(t) = e^{i\Omega_1 t} + e^{i\Omega_2 t}, \tag{5.9}$$

$$x^{(b)}(t) = e^{i\Omega_3 t}, \tag{5.10}$$

$$\begin{aligned} y^{(p)}(t) = {} & H_1^{(p:a)}(\Omega_1)e^{i\Omega_1 t} + H_1^{(p:a)}(\Omega_2)e^{i\Omega_2 t} + H_1^{(p:b)}(\Omega_3)e^{i\Omega_3 t} + 2H_2^{(p:aa)}(\Omega_1,\Omega_2)e^{i(\Omega_1+\Omega_2)t} \\ & + 2H_2^{(p:ab)}(\Omega_1,\Omega_3)e^{i(\Omega_1+\Omega_3)t} + 2H_2^{(p:ab)}(\Omega_2,\Omega_3)e^{i(\Omega_2+\Omega_3)t} \\ & + 6H_3^{(p:aab)}(\Omega_1,\Omega_2,\Omega_3)e^{i(\Omega_1+\Omega_2+\Omega_3)t} + \cdots . \end{aligned} \tag{5.11}$$

A much more detailed discussion of MIMO harmonic probing can be found in [36].

## (b) Multi-degree-of-freedom NARX neural network structure

The MIMO case also requires an extension of the NARX model structure from equation (4.7); one needs a structure which regresses each output on multiple inputs; that is,

$$y_t^{(p)} = F^{(p)}(y_{t-1}^{(1)}, \cdots, y_{t-n_y}^{(1)}, x_t^{(1)}, \cdots, x_{t-(n_x-1)}^{(1)}, \cdots, y_{t-1}^{(n)}, \cdots, y_{t-n_y}^{(n)}, x_t^{(n)}, \cdots, x_{t-(n_x-1)}^{(n)}). \tag{5.12}$$

Although this expression looks more complex, it is simply a multivariate function as before, just with more arguments. Assuming that one attempts to fit the output variables one at a time, one could use the GP formulation as before; however, machine-learning models exist which can learn full MIMO systems. Because the multi-output GP is a little more demanding, the model used here is the standard multi-layer perceptron (MLP) neural network [2].In fact, the first paper on HFRFs from machine learning [13] used an MLP structure.

Harmonic probing on the NARX model structure can recover all the different HFRFs between different points and at different nonlinear orders; however, it is necessary to establish the correct notation for the task. Assuming a sigmoidal neural network, the MLP outputs can be expanded as polynomials in the inputs; for output $p$, over nonlinear orders $i$, one has,

$$y_t^{(p)} = \sum_{j=0}^{n_h} \frac{w_j^{(p)} \tanh(b_j)^{(i)}}{i!} \left( \sum_{m=0}^{n_x-1} u_{jm}^{(1)} x_{t-m}^{(1)} + \cdots + \sum_{m}^{n_x-1} u_{jm}^{(n)} x_{t-m}^{(n)} + \sum_{k=1}^{n_y} v_{jk}^{(1)} y_{t-k}^{(1)} + \cdots + \sum_{k=1}^{n_y} v_{jk}^{(n)} y_{t-k}^{(n)} \right)^i, \tag{5.13}$$

where the weights $w_j^{(p)}$ connect the MLP hidden layer to output node $p$, and the weights between inputs and hidden layer are grouped into the $u$ and $v$ variables; the weights $b_j$ connect the hidden units to a bias node and $\sigma$ is a standard sigmoid activation function. The general positions of the various weights are presented in the schematic in figure 13.

The number of lags for each input and output lag was set to a common value; this simplified the lag selection optimization considerably and did not substantially degrade the results in this case.

## (c) A MIMO case study

To illustrate the MIMO approach here in the simplest situation, a two degree-of-freedom (2DOF) nonlinear lumped-mass system was chosen, as shown in figure 14, with equations of motion:

$$m\ddot{y}_1 + c\dot{y}_1 + c(\dot{y}_1 - \dot{y}_2) + ky_1 + k(y_1 - y_2) + k_2{y_1}^2 + k_3{y_1}^3 = x_1(t), \tag{5.14}$$

$$m\ddot{y}_2 + c\dot{y}_2 + c(\dot{y}_2 - \dot{y}_1) + ky_2 + k(y_2 - y_1) + k_2{y_2}^2 + k_3{y_2}^3 = x_2(t). \tag{5.15}$$

As a sanity check, one can apply harmonic probing to these equations directly; for the first-order kernel, the probing inputs required are $x^{(p)}(t) = e^{i\Omega t}, x^{(q)} = 0$, leading to outputs of $y^{(p)}(t) =$

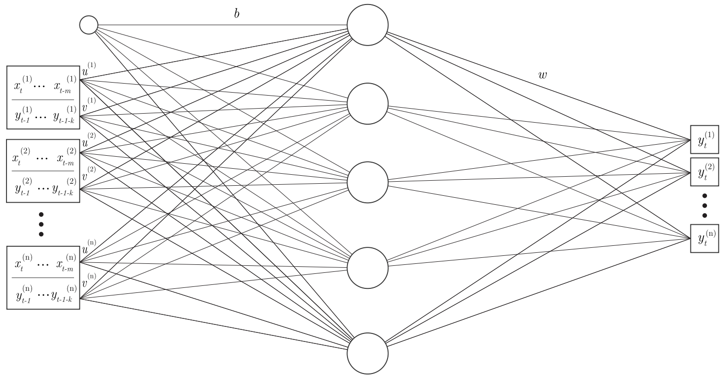
b
w
$x_t^{(1)} \cdots x_{t-m}^{(1)}$
$y_{t-1}^{(1)} \cdots y_{t-1-k}^{(1)}$
$u^{(1)}$
$v^{(1)}$
$x_t^{(2)} \cdots x_{t-m}^{(2)}$
$y_{t-1}^{(2)} \cdots y_{t-1-k}^{(2)}$
$u^{(2)}$
$v^{(2)}$
$x_t^{(n)} \cdots x_{t-m}^{(n)}$
$y_{t-1}^{(n)} \cdots y_{t-1-k}^{(n)}$
$u^{(n)}$
$v^{(n)}$
$y_t^{(1)}$
$y_t^{(2)}$
$y_t^{(n)}$

**Figure 13.** MIMO NARX MLP neural network structure.

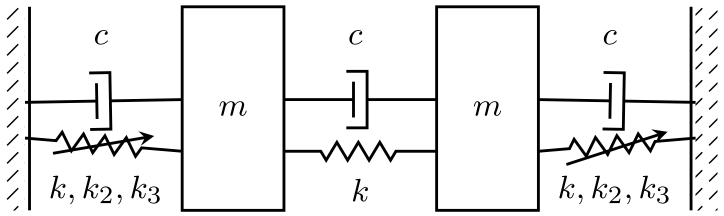

**Figure 14.** 2DOF case study system.

$H_1^{(p:p)}(\Omega)e^{i\Omega t}$ and $y^{(q)}(t) = H_1^{(q:p)}(\Omega)e^{i\Omega t}$. Substituting into equation (5.15), one arrives at the linear FRF matrix,

$$\mathbf{H_1}^{(2)}(\Omega) = \begin{bmatrix} H_1^{(1:1)}(\Omega) & H_1^{(1:2)}(\Omega) \\ H_1^{(2:1)}(\Omega) & H_1^{(2:2)}(\Omega) \end{bmatrix} = \begin{bmatrix} -m\Omega^2 + 2k + 2ci\Omega & -ci\Omega - k \\ -ci\Omega - k & -m\Omega^2 + 2k + 2ci\Omega \end{bmatrix}^{-1}, \quad (5.16)$$

as expected.

In practice, the objective is to fit a MIMO NARX model to system data and probe the NARX model to estimate the HFRFs. The harmonic probing expressions for the full MIMO are very large and complicated; their forms are not given here, but will be postponed to a future publication. However, the results of that process for data simulated from the 2DOF case study, are given.

The parameters for simulation were chosen as $m = 1kg$, $c = 5Ns^{-1}$, $k = 1 \times 10^4$, $k_2 = 1 \times 10^7$ and $k_3 = 5 \times 10^9$. To train the NARX neural network, output data were generated over 20 s with time step 0.0005 s using a fourth-order Runge–Kutta integration scheme [26]. The data were then sub-sampled to a time step of 0.0025 s (sampling frequency of 400Hz). The system was forced using a multisine input with r.m.s. 2N with a bandwidth of (0–50Hz) to replicate Gaussian white noise. In the 2DOF system chosen there are certain symmetries, so there are relations between the kernels; for example, at first-order $H_1^{(1:1)} = H_1^{(2:2)}$ and $H_1^{(1:2)} = H_1^{(2:1)}$. For ease of visualization, these repeating kernels are not displayed here.

The data were then fitted by the MIMO MLP model using Pytorch. Adam gradient was used with learning rate = $1 \times 10^{-2}$and a single hidden layer with 60 neurons. Four lags were taken for all input and output variables. Using the weights from the trained model, the HFRFs were extracted using MIMO harmonic probing. As the continuous-time equations of motion were known, they could be used a ground truth for comparison. The exercise conducted here used a noise-free training set of input and output data, the NARX model gave model-predicted NMSE values of 0.02 and 0.04 on validation and testing sets, respectively; this shows an excellent fit. Figures 15 and 16 show the comparisons between the HFRFs and ground truth (the $H_2$ diagonals are shown); the results are very good, as one might expect given the model fit.

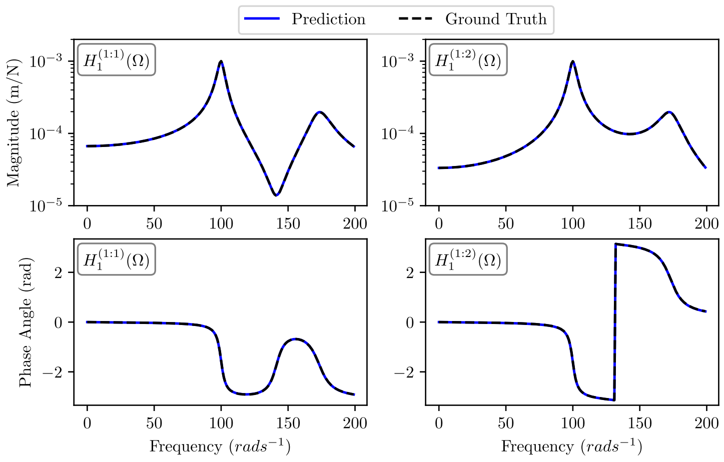

**Figure 15.** Comparison of $H_1$ kernel predictions with ground truth for 2DOF system.

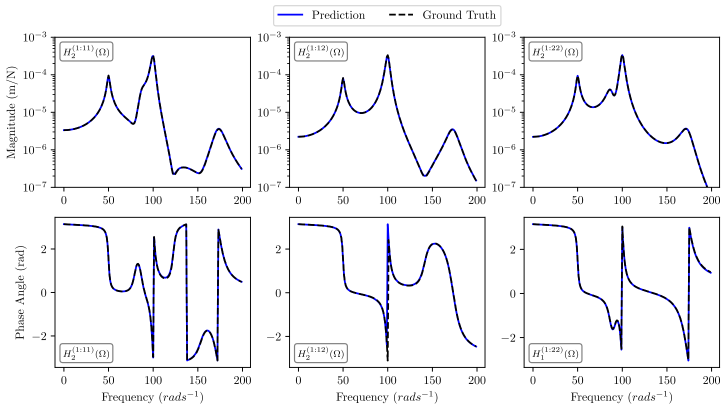

**Figure 16.** Comparison of $H_2$ kernel predictions diagonal with ground truth for 2DOF system.

## 6. Conclusions

Long conclusions are not warranted here. The aim of the paper has been to show that developments in machine learning over the last two or three decades have proved very effective in Volterra-series estimation, both for kernels themselves and for the HFRFs. A number of technologies are presented here; in the first case, an RKHS approach is shown, which estimates discrete kernels directly from a form of regularised least-squares analysis. Second, a GP-NARX approach is presented, which allows direct estimation of HFRFs. Finally a neural network NARX model is shown to be effective in estimating HFRFs for MIMO systems. In some ways, the exact machine-learning algorithms are not so important; in each case, the main objective is to fit a predictive time-series model. However, in the case of HFRFs for example, one would need to perform a harmonic probing calculation appropriate to whatever model basis is chosen. The ones shown here have been calculated for a Gaussian basis—appropriate to a GP or a radial-basis function network—and for a sigmoidal neural network structure; these are arguably the most common bases. The idea here has been to provide an overview of the main ideas, although some new results have been included here in terms of the MIMO results and also a 'simple' demonstration that the Volterra series is a functional Taylor series.

Data accessibility. Supplementary material is available online [37].
Declaration of AI use. We have not used AI-assisted technologies in creating this article.

Authors' contributions. K.W.: conceptualization, funding acquisition, investigation, methodology, software, supervision, writing—original draft; T.R.: conceptualization, formal analysis, funding acquisition, methodology, supervision, writing—original draft; O.P.: formal analysis, methodology, software, writing—original draft.

All authors gave final approval for publication and agreed to be held accountable for the work performed therein.

Conflict of interest declaration. We declare we have no competing interests.

Funding. O.J.P. would like to acknowledge support for his PhD studentship from Schlumberger; K.W. would like to thank the EPSRC for support via an Open Fellowship (EP/X040852/1).

Acknowledgements. For the purpose of open access, the authors have applied a Creative Commons Attribution (CC BY-ND) licence to any Author Accepted Manuscript version arising.

## Endnotes

[1]Unfortunately, the term 'non-parametric' can be used in different ways; the sense here is from the structural dynamics community, regarding lack of physical meaning of the parameters. This use is not so far from the usual sense in machine learning. However, one should note that some machine learning users refer to a model as non-parametric if it predicts using the training data [3].

[2]Apart from being an outstanding mathematician, Volterra appears to have been a man of principle; in 1931, at the end of a distinguished career, he refused to take an oath of loyalty to the fascist government; as a result, he was removed from his university chair and expelled from the Italian scientific academies.

[3]Note that there are restrictions on the type of functionals which admit a Volterra representation. In particular, the functionals of interest must be *single-valued*; this means that systems which bifurcate under certain conditions—like the Duffing oscillator of equation (1.1)—do not have a Volterra series for the system under those conditions. More generally, it is often said that the Volterra series does not exist if a system exhibits *strong nonlinearity*. The qualifier 'strong' is rarely precisely defined, but one operational definition would be to say that this means that bifurcations are excluded, along with associated behaviour like subharmonics or chaos; one might say that a system is weakly nonlinear if only superharmonics can appear in the response. A more detailed discussion of the requirements for a Volterra series to exist can be found in [19]. A more difficult problem arises when one asks questions of whether the series *converges*. This is a very difficult question, as concepts like 'radius of convergence' need to extend to function spaces. In fact, there are very few convincing studies of convergence, mostly these are restricted to specific systems and specific inputs and are simply observational. Perhaps one of the most convincing analyses was provided by Barrett, early in the applications to engineering [9]. Again, as an observation, it is noted in [1], that a truncated Volterra expansion begins to lose accuracy as the system *approaches* a bifurcation; the context there is where the frequency of a harmonic input approaches a jump frequency in the FRF of a Duffing oscillator. Of course, this often limits the use of the Volterra series to qualitative analysis, but this can be valuable in itself.

[4]The tedious part is the bookkeeping; i.e. keeping track of the order of the Volterra coefficients in the parameters $w_k$.

[5]Note that this is a speed-up in prediction on new points; as will be shown later, estimating the coefficients is an $O(D^3)$ problem.

[6]In fact, the data for this section and the following were simulated with a *time step* of 0.001 s and then downsampled. This idea balanced the accuracy of the simulation with the number of points of training data. Previous experience with many parametric identification methods showed that this sampling frequency gives good results for the systems of interest.

[7]At first glance, this looks like a very small number; however, consider the following approximate argument. The regularization constant can be regarded as an estimate of the noise variance $\sigma_n^2$ (see next section). In this case, $\sigma_n = 0.1\sigma_y$, where $\sigma_y$ is the s.d. of the response. Now, very approximately $\sigma_y \approx \sigma_x/k$, so in this case $\sigma_n \approx 10^{-5}$.

[8]In fact, this is an issue also faced by the RKHS approach.

[9]The reader will have noticed that equations (4.11) and (4.12) bear a striking resemblance to equations (3.20) and (3.33) from the RKHS treatment in the previous section. In fact, one can show that the GP and RKHS approaches are very closely related, as discussed in chapter 6 of [27].

## References

1. Worden K, Tomlinson GR. 2001 *Nonlinearity in structural dynamics*. Bristol, UK: Institute of Physics Press.
2. Bishop CM. 2013 *Pattern recognition and machine learning*. New York, NY: Springer.

3. Lindholm A, Wahlström N, Lindsten F, Schön TB. 2022 *Machine learning: a first course for engineers and scientists*. Cambridge, UK: Cambridge University Press.
4. Volterra V. 1959 *Theory of functionals and integral equations*. Garden City, NY: Dover Publications.
5. Schetzen M. 1980 *The volterra and wiener theories of nonlinear systems*. Hoboken, NJ: John Wiley Interscience Publication.
6. Volterra V. 1887 Sopra le funzioni che dipendono de altre funzioni. *Rend R Acad. Dei Lincei* **2**, 141–146.
7. Marmarelis PZ, Naka KI. 1974 Identification of multi-input biological systems. *IEEE Trans. Biomed. Eng*. **21**, 88–101. (doi:10.1109/TBME.1974.324293)
8. Barrett JF. 1963 The use of functionals in the analysis of non-linear physical systems. *J. Electron. Control* **15**, 567–615. (doi:10.1080/00207216308937611)
9. Barrett JF. 1965 The use of volterra series to find the region of stability of a non-linear differential equation. *Int. J. Control* **1**, 209–216. (doi:10.1080/00207176508905473)
10. Gifford SJ, Tomlinson GR. 1989 Recent advances in the application of functional series to non-linear structures. *J. Sound Vib*. **135**, 289–317. (doi:10.1016/0022-460x(89)90727-x)
11. Billings SA, Tsang KM. 1989 Spectral analysis for non-linear systems, part I: parametric non-linear spectral analysis. *Mech. Syst. Signal Process*. **3**, 319–339. (doi:10.1016/0888-3270(89)90041-1)
12. Wray J, Green GGR. 1994 Calculation of the Volterra kernels of non-linear dynamic systems using an artificial neural network. *Biol. Cybern*. **71**, 187–195. (doi:10.1007/s004220050081)
13. Chance JE, Worden K, Tomlinson GR. 1998 Frequency domain analysis of NARX neural networks. *J. Sound Vib*. **213**, 915–941. (doi:10.1006/jsvi.1998.1539)
14. Dodd TJ, Harrison RF. 2003 Estimating Volterra filters in Hilbert space. In *Proc. IFAC Int. Conf. on intelligent control systems and signal processing*.
15. Wan Y, Dodd TJ, Harrison RF. 2005 Identification of infinite degree Volterra series in the time and frequency domains. In *Proc. of the 15th IFAC World Congress*, Prague.
16. Dodd TJ, Wan Y, Drezet P, Harrison RF. 2007 Practical estimation of Volterra filters of arbitrary degree. *Int. J. Control* **80**, 908–918. (doi:10.1080/00207170701216303)
17. Worden K, Becker WE, Rogers TJ, Cross EJ. 2018 On Gaussian process NARX models and their higher-order frequency response functions: an application to wave force prediction. *Mech. Syst. Signal Processing* **104**, 188–223. (doi:10.1016/j.ymssp.2017.09.032)
18. Worden K, Surace C, Becker W. 2017 Uncertainty bounds on higher-order FRFs from Gaussian process NARX models. In *Eurodyn X– 10th Int. Conf. on Structural Dynamics*, Rome, Italy.
19. Palm G, Poggio T. 1977 The Volterra representation and the wiener expansion: validity and pitfalls. *SIAM J. Appl. Math*. **33**, 195–216. (doi:10.1137/0133012)
20. Nash C. 2010 *Relativistic quantum fields*. Garden City, NY: Dover Publications.
21. Aronszajn N. 1950 Theory of reproducing kernels. *Trans. Am. Math. Soc*. **68**, 337–404. (doi:10.1090/s0002-9947-1950-0051437-7)
22. Schölkopf B, Smola AJ, Müller K. 2002 *Learning with kernels*. Cambridge, MA: MIT Press.
23. Dodd TJ, Harrison RF. 2003 A new solution to Volterra series estimation. InIFAC 15th triennial world congress. In *IFAC 15th Triennial World Congress*, Barcelona, Spain.
24. Birpoutsoukis G, Marconato A, Lataire J, Schoukens J. 2017 Regularized nonparametric Volterra kernel estimation. *Automatica* **82**, 324–327. (doi:10.1016/j.automatica.2017.04.014)
25. Birpoutsoukis G, Csurcsia PZ, Schoukens J. 2018 Efficient multidimensional regularization for Volterra series estimation. *Mech. Syst. Signal Process*. **104**, 896–914. (doi:10.1016/j.ymssp.2017.10.007)
26. Press WH, Teukolsky SA, Vetterling WT, Flannery BP. 1992 *Numerical recipes in C*. Cambridge, UK: Cambridge University Press.
27. Rasmussen CE, Williams CKI. 2006 *Gaussian processes for machine learning*. Cambridge, MA: MIT Press. (doi:10.7551/mitpress/3206.001.0001)
28. Leontaritis IJ, Billings SA. 1985 Input-output parametric models for non-linear systems Part I: deterministic non-linear systems. *Int. J. Control* **41**, 303–328. (doi:10.1080/0020718508961129)
29. Leontaritis IJ, Billings SA. 1985 Input-output parametric models for non-linear systems Part II: stochastic non-linear systems. *Int. J. Control* **41**, 329–344. (doi:10.1080/0020718508961130)
30. Billings SA. 2013 *Nonlinear system identification: NARMAX, methods in the time, frequency, and spatio-temporal domains*. Hoboken, NJ: Wiley-Blackwell.

31. Quinonero-Candelo J, Rasmussen CE. 2005 A unifying view of sparse approximation Gaussian process regression. *J. Mach. Learn. Res.* **6**, 1939–1959.
32. Snelson E, Ghahramani Z. 2006 Sparse Gaussian processes using pseudo-inputs. In *Advances in neural information processing systems* (eds Y Weiss, B Schölkopf, J Platt). Cambridge, MA: MIT Press.
33. Giraud A. 2004 Approximate methods for propagation of uncertainty with gaussian process models. PhDthesis, University of Glasgow.
34. Quinonero-Candelo J, Giraud A, Rasmussen CE. 2003 *Prediction at an uncertain input for Gaussian processes and relevance vector machines – application to multiple-step ahead time series forecasting, Technical report*. Department of Informatics and Mathematical Modelling, Technical University of Denmark.
35. Bedrosian E, Rice SO. The output properties of Volterra systems (nonlinear systems with memory) driven by harmonic and Gaussian inputs. *Proc. IEEE* **59**, 1688–1707. (doi:10.1109/PROC.1971.8525)
36. Worden K, Manson G, Tomlinson GR. 1997 A harmonic probing algorithm for the multi-input Volterra series. *J. Sound Vib.* **201**, 67–84. (doi:10.1006/jsvi.1996.0746)
37. Worden K, Rogers T, Preston O. 2025 Supplementary material from: Machine-learning perspectives on Volterra system identification. Figshare (doi:10.6084/m9.figshare.c.8039942)