Chapter 8
A Tutorial on Data-Driven Methods in Nonlinear Dynamics

Keith Worden and Elizabeth J. Cross

Abstract In the past two decades, it is fair to say that there has been an explosion in the use of machine learning technology or 'data-driven' methods, across the whole subject of engineering; this is no less true of the subdiscipline of structural dynamics. A modern dynamicist needs, at the least, some familiarity with these technologies. This paper attempts to give an overview of some of the main ideas in 'data-based' engineering, by focussing on the (comparatively) smaller area of nonlinear dynamics—indeed on nonlinear system identification. A particular viewpoint is adopted, based on modern Bayesian methods of regression. Considerable attention is paid here to the desirability of combining measured data with physical insight when modelling dynamic systems and structures. Although this view naturally begins with the idea of 'grey-box' models, this generalises into the emerging subject of *physics-informed machine learning*. Although this tutorial necessarily focusses down on a narrow application context, the many references allow the curious reader to explore further afield.

Keywords Tutorial · Data-based methods · Nonlinear dynamics

8.1 Introduction

At the risk of beginning this tutorial paper with a contentious statement, one might argue that *all* of science and engineering is *data-based*. It is certainly true that the acid test of any physical theory is that it be consistent with previous observations (data), and that it make predictions that can be confirmed by further new observations. As all of science and engineering is arguably founded in physics, the opening statement follows. Adopting the term 'data-driven' narrows the field a little; however, even then the subject is far too large to address in a single paper. Focussing down further, this paper is concerned with 'data-driven methods', and in the recent past, this term has come to mean a specific class of algorithms derived from statistics and machine learning, with which one can model some aspect of reality armed with some prior assumptions and a set of measured observations from the system of interest. Finally, the paper will concentrate on the specific context of nonlinear dynamics. With this remit, one might arguably do justice to the subject matter with a single book; the recent [1] is a nice attempt. However, this is a paper, not a book, so it will reflect some hard choices on the part of the authors.

The term 'model' here will be applied to any *mathematical* construct which can be used to explain observations of physical reality and predict the observations that would entail for the system under previously unobserved stimuli. The discipline of *machine learning* will be understood to mean the body of theory and practice which allows one to infer a model from some subset of the known observations, possibly augmented by some prior knowledge. To establish some concrete terminology here, any model derived entirely from the data alone will be referred to as a 'black-box' model. If a model has been derived purely by the application of existing laws of physics, it will be referred to as a 'white-box' model.[1] A model, which is inferred from prior physics *and* observed data, will be called a 'grey-box' model.

Of course, learning a model from data has been an established discipline within engineering for decades—that of *system identification* (SI). As such, the overall problem breaks down into two essential components *structure detection* and

[1] This definition will gloss over the fact that 'existing' laws of physics will, at some point, have been based on observational evidence (data).

K. Worden (✉) · E. J. Cross
Dynamics Research Group, Department of Mechanical Engineering, University of Sheffield, Sheffield, UK
e-mail: k.worden@sheffield.ac.uk; e.j.cross@sheffield.ac.uk

M. R. W. Brake et al. (eds.), *Nonlinear Structures & Systems, Volume 1*, Conference Proceedings of the Society for Experimental Mechanics Series, https://doi.org/10.1007/978-3-031-36999-5_8

parameter estimation.[2] A ‘model’ is defined above as a ‘mathematical construct’; often this means a mathematical equation, or system of equations. With this in mind, the problem of structure detection is simply that of determining the functional form of the equations which govern the model; they may be algebraic, differential, integral, etc. The equations will need to contain all of the variables which control or describe some aspect of reality (temperature, current, acceleration, etc.) and may well contain constants or parameters, some of which will be fixed by the physics and some will need to be determined. The latter parameters give rise to the problem of parameter estimation and will need to be inferred from measured data. As a concrete example (which will recur throughout the paper), one might consider a dynamical system of the form:

$$m\ddot{y} + c\dot{y} + ky = x(t) \tag{8.1}$$

which is a simple linear mass-spring-damper system. In the usual frame of engineering dynamics, $x(t)$ would be a force (the stimulus), and $y(t)$ would be a displacement (the response); m, c and k would be constants for a given system (mass, damping and stiffness, respectively). The overdots in the equation denote differentiation with respect to time; thus the model *structure* here represents a second-order differential equation. If the structure and values of the constants are known a priori from physics, this is a white-box model; if the structure is known, but the parameters need to be inferred from measured data, this is a *grey-box* model.

The problem of inferring model parameters from data is commonly called *regression* within the statistics and machine learning communities [2]. Few would argue that the mathematical basis for regression began with the method of *least squares*. This algorithm for fitting a model to data was first published by Legendre [3], although Gauss subsequently claimed precedence.[3] Regardless of the dispute on precedence, Gauss certainly extended the algorithm in important ways; in particular, the *Gaussian* or *normal* probability distribution was introduced as a means of handling measurement errors in a rigorous fashion. In any case, both Legendre and Gauss developed and used the method in order to solve problems in celestial mechanics, which were highly nonlinear. The point here is that it would appear that the discipline of machine learning was originally motivated by problems in nonlinear dynamics.

Given that the subject of ‘data-driven’ methods is vast, the ‘hard choices’ mentioned earlier come into play. Given the experience and preferences of the authors, this tutorial will concentrate on system identification. Furthermore, as this paper is a tutorial as opposed to original research, it will make no attempt at a comprehensive or balanced survey of the literature, but will rather lean on the papers by the authors and their colleagues with which they are most familiar. A more balanced viewpoint can be found by following the references in those papers.

The identification of linear systems (LSI) is arguably so well developed now that it is comprehensively covered in textbooks [5, 6]. In contrast, nonlinear system identification (NLSI) is by no means as well developed [7]. Part of the problem with NLSI is that the structure detection problem is much more complicated. For a general linear differential equation model, the only freedom in the model structure is in the number of derivatives of the variables; furthermore, the model parameters only appear as multipliers of terms, there are no parameters ‘hidden’ in nonlinear functions, like the a in e^{ay}. Such parameters require more sophisticated means of estimation, whereas the *linear-in-the-parameters* problems can sometimes yield to algorithms as ‘simple’ as basic least squares. An NLSI model structure could in principle contain *any* nonlinear functions of the variables of interest and their derivatives and any parameters. The complexities of NLSI have meant that, in the past, there has been no single algorithm which can address the completely general problem; the NLSI practitioner has instead relied on a ‘toolbox’ philosophy, with different approaches used for different classes of problem.

This situation persisted until recently, when two methodologies emerged in the NLSI community, each offering the prospect of a general framework. The first group of algorithms, based on *evolutionary optimisation*, starting with the genetic algorithm [8], can handle difficult technical problems in NLSI, like nonlinearity in the model parameters and the existence of unmeasured states [9, 10]. The second group exploited *Bayesian Inference*. Although this idea originated over 20 years ago [11, 12], it really flourished in the last decade or so, when dynamicists began to take advantage of concepts from machine learning [13]. Bayesian methods overcome the same technical problems as evolutionary methods [14] and offer additional benefits; they allow model selection simultaneously with parameter estimation [15], can estimate parameter *distributions* and can propagate uncertainty in a principled manner. Because of its power and generality, this tutorial will mainly focus on Bayesian methods.

Having said that the focus here will be on NLSI, it is important to note that there are many other problems in nonlinear dynamics for which a data-driven approach has been adopted. In fact, data-driven methods have proved vital in the development of the subject for a fundamental and important reason. Problems in nonlinear dynamics almost never have

[2] The first author first learned these terms from the seminal work of Steve Billings; however, they may well predate that work.

[3] Legendre published in 1805, Gauss in 1809; however, Gauss claimed to have had the method since 1793. The whole story is discussed in [4].

analytical solutions. To be more precise, nonlinear equations of motion almost never have exact closed-form solutions. Suppose one were to take the linear equation in Eq. (8.1) and add the 'simplest' nonlinear term (which turns out to be cubic in y), the result is *Duffing's equation* [16]:

$$m\ddot{y} + c\dot{y} + ky + k_3 y^3 = x(t) \tag{8.2}$$

To this day, there are no exact solutions of this equation, except in the very restricted undamped ($c = 0$) and unforced ($x(t)$) case. For the restricted case, Duffing himself provided an exact solution involving elliptic functions in 1918; however, there has been no progress to speak of on the general case since then. Unfortunately, this general lack of exact solutions means that all of the fascinating nonlinear phenomena discovered in the twentieth century—like bifurcations and chaos—present intractable problems for exact analytical methods. One way around this issue has been to rely on simulations; one numerically simulates the system responses of interest and characterises the responses. Initially these simulations were carried out using analogue computers incorporating bespoke nonlinear circuits, but with the advent of (and explosion in) digital computers, it became possible to solve the nonlinear initial value problems in order to simulate samples of response data for further computational analysis. For example, if one wishes to analyse the stability or chaotic nature of a system, the Lyapunov exponents can be estimated from a time series [17] (in fact, the algorithm in that paper was designed for use with experimental data, but the principle is the same). Discrete wavelet analysis has also proved powerful in the analysis of chaotic time series [18]; apart from estimating certain chaotic invariants, the analysis can also detect coherent structures. Finally, on the subject of coherent structures, the interesting book [19] details how coherent structures and low-dimensional structure can be found in data from turbulent flows; one of the main algorithms discussed is the Karhunen-Loéve expansion or principal orthogonal decomposition (POD). Perhaps a little ironically, the POD is basically a variant of principal component analysis (PCA), which is a *linear* projection method often used in applied machine learning for data visualisation and dimension reduction. Turbulence is of course one of the major outstanding problems in nonlinear science generally.

Even within the restricted scope of NLSI, this paper will omit one very important class of models—the 'black-box' models mentioned earlier. In principle, such models can work without any physical insight whatsoever; they work by proposing some basis of functions which spans the function space of interest, much as a Fourier expansion does for the space of periodic functions on some finite interval. The important point of such models is not that they encode some prior physical knowledge, but rather that they have a *universal approximation property*, which is to say that they can represent some target function with arbitrary accuracy (as long as they include enough terms). Such 'learners' include neural networks (deep or shallow), radial-basis function (RBF) networks, Gaussian processes, etc. [2]. In principle, one could download software for such a learner, input some data and run the software with default settings, and out would pop a model. Such a model would be 'pitch black'.[4] The authors would always caution against this practice; it seldom produces good research and could produce catastrophic results when applied in real life. Black boxes will surface later in the discussion of grey boxes and *physics-informed machine learning* (PIML).

The layout of the paper is as follows. The next section presents the arguments for adopting a Bayesian approach to NLSI and this is followed by a discussion of a specific parameter estimation method based around Markov chain Monte Carlo (MCMC). Section 8.4 then presents a case study of MCMC applied to the identification of a Duffing oscillator system. Section 8.5 discusses the combined problem of parameter estimation and model selection and is followed by another case study involving equation discovery for hysteretic oscillators. Section 8.7 then presents some ideas on combining physics-based and data-based approaches in the form or grey-box modelling, or more generally *physics-informed machine learning*. The paper ends with brief conclusions.

8.2 Bayesian Inference and System Identification

This section will give a precise definition of the system identification problem and will outline the advantages of taking a Bayesian probabilistic viewpoint. The input (stimulus) variable will be denoted $x(t)$ and the output (response) variable by $y(t)$. Suppose that one has a set of data $\mathcal{D} = \{(x_i, y_i), i = 1, \dots, N\}$ of sampled system inputs x_i and outputs y_i. If one assumes that there is no measurement noise, and the model structure is known, then the application of an identification algorithm will yield (assuming that the problem is well-conditioned) a deterministic estimate of the system parameters $\underline{w}$:

$$\underline{w} = id(\mathcal{D}) \tag{8.3}$$

[4] The authors thank Johan Schoukens for introducing them to this term.

where the function id represents the application of the identification algorithm to the data $\mathcal{D}$. If one were concerned with the Duffing system of Eq. (8.2), then one would have $\underline{w} = (m, c, k, k_3)^T$. Now, if noise $\epsilon(t)$ is present on the input or output data (or both), $\underline{w}$ will become a *random variable* conditioned on the data. In this context one no longer wishes to find an *estimate* of $\underline{w}$, but rather to specify one's belief in its value. To simplify matters here, it will be assumed that the noise is Gaussian with (unknown) variance σ_ϵ; also, the parameter σ_ϵ will be subsumed into $\underline{w}$, since it is to be inferred along with the model. In probabilistic terms, instead of Eq. (8.3) one now has:

$$\underline{w} \sim p(\underline{w}|\mathcal{D}, \mathcal{M}) \tag{8.4}$$

where $\mathcal{M}$ represents the choice of model (the Duffing oscillator in the current case). Knowing the full parameter distribution means that one can construct confidence intervals (or error bars) for the parameter estimates; this is the first major advantage of a probabilistic viewpoint. In particular, the *Bayesian* approach to SI is that one uses Bayes' theorem in the form:

$$p(\underline{w}|\mathcal{D}) = \frac{p(\mathcal{D}|\underline{w})p(\underline{w})}{p(\mathcal{D})} \tag{8.5}$$

to convert an a priori probability distribution for the parameters $\underline{w}$ into a posterior distribution having seen the data $\mathcal{D}$. If one desires a so-called point estimate of the parameters, the usual course of action is to choose that which maximises the posterior probability $p(\underline{w}|\mathcal{D})$. Now, as the data $\mathcal{D}$ are a constant of the identification problem, one is reduced to maximising $p(\mathcal{D}|\underline{w})p(\underline{w})$. It is often the case at this point that an uninformative constant (and hence improper) prior $p(\underline{w})$ is chosen, and this reduces the problem to that of maximising $p(\mathcal{D}|\underline{w})$, which is simply the likelihood of the data. The *maximum a posteriori* (MAP) estimate then becomes *maximum likelihood*. If the noise is assumed Gaussian, the problem essentially becomes one of minimising a least-squares cost/error function.

The usual objective of SI is to provide a *predictive model*, i.e. one which can estimate or predict system outputs if a different system input were provided. In the probabilistic context described above, the best that one could do is to determine a *predictive distribution*. Suppose a new input sequence $\underline{x}^*$ were applied to the system, one would wish to determine the density for the predicted outputs:

$$\underline{y}^* \sim p(\underline{y}^*|\underline{x}^*, \underline{w}, \mathcal{D}, \mathcal{M}) \tag{8.6}$$

noting all the dependencies.

The mean of this distribution would give the 'best' estimates for the predictions and the covariance would allow one to establish confidence intervals for them. However, one notes the presence of the parameter vector $\underline{w}$. In practice, one might use the $\underline{w}$ value corresponding to the mean or the mode of the posterior parameter distribution; however, a truly Bayesian viewpoint on the prediction would require one to marginalise over the parameter estimates, i.e. to derive:

$$p(\underline{y}^*|\underline{x}^*, \mathcal{D}, \mathcal{M}) = \int p(\underline{y}^*|\underline{x}^*, \underline{w}, \mathcal{M})p(\underline{w}|\mathcal{D}, \mathcal{M})d\underline{w} \tag{8.7}$$

This is a very powerful idea: allowing for a fixed model *structure*, *one is making predictions using an entire set of parameters consistent with the training data*, with each point in the space of parameters weighted according to its probability given the data. In practice, there are considerable problems in implementing the full Bayesian approach, i.e. performing the intractable integral 8.7. For the purposes of this section, the main issue for discussion will be the problem of inferring the distributions for the parameters as given in Eq. (8.4).

Another potential advantage of a Bayesian approach is that it may be possible to assess the relative evidence for a number of competing model structures, i.e. to solve the *structure detection* problem. Suppose that one believes that the true model structure is one of a finite number $\{\mathcal{M}_i, i = 1, \ldots, M\}$ (the discussion here will closely follow [13]). In principle, one could imagine computing the probability of observing the data $P(\mathcal{D}|\mathcal{M}_i)$, given the particular model structure (noting that this is a from a discrete distribution). If this quantity were available, one could select the model with the highest probability. Even more in the spirit of Bayesian inference, one could marginalise over *all possible* model structures weighted according to their probability; in terms of prediction, one would have:

$$p(\underline{y}^*|\underline{x}^*, \mathcal{D}) = \sum_{i=1}^{M} p(\underline{y}^*|\underline{x}^*, \mathcal{M}_k, \mathcal{D})P(\mathcal{M}_k|\mathcal{D}) \tag{8.8}$$

Unfortunately, the posterior over models $P(\mathcal{M}_i|\mathcal{D})$ is difficult to compute. If one appeals to Bayes theorem in the form:

$$P(\mathcal{M}_i|\mathcal{D}) = \frac{p(\mathcal{D}|\mathcal{M}_i)P(\mathcal{M}_i)}{p(\mathcal{D})} \tag{8.9}$$

and assumes equal priors on the models, one arrives at a comparison ratio or *Bayes factor*:

$$B_{ij} = \frac{P(\mathcal{M}_i|\mathcal{D})}{P(\mathcal{M}_j|\mathcal{D})} = \frac{p(\mathcal{D}|\mathcal{M}_i)}{p(\mathcal{D}|\mathcal{M}_j)} \tag{8.10}$$

which weights the evidence for two models in terms of marginal likelihoods of the data given the models. Unfortunately, the marginal likelihoods themselves are expressed as an integral over all possible parameters, i.e.

$$p(\mathcal{D}|\mathcal{M}) = \int p(\mathcal{D}|\underline{w}, \mathcal{M})p(\underline{w}|\mathcal{M})d\underline{w} \tag{8.11}$$

and this integral is analytically intractable and numerically challenging because it can be high-dimensional [20].

One can resort to less informative model selection indicators which are simpler to compute, e.g., the *deviance information criterion* (DIC) [21], as applied in [14].

Leaving aside model selection for the moment, the next section will discuss a Bayesian approach to parameter estimation—*Markov chain Monte Carlo* (MCMC), a powerful approach for sampling from probability distributions.

8.3 System Identification Using Markov Chain Monte Carlo

As stated in the previous section, one of the central problems discussed in this paper is that of determining the posterior probability function for parameters, given data from a differential equation system. This distribution will not take any particular form (e.g. Gaussian) and in the absence of the normalising constant $p(\mathcal{D}|\mathcal{M})$ can only be evaluated up to proportionality. One ideally wishes to adopt a nonparametric approach to the problem so that the true statistics of the parameter density are obtained. An obvious step in the direction of nonparametric estimation is to sample from the parameter distribution itself and thus build a picture; however, the distribution in question is unknown. Fortunately, there exists a very powerful class of numerical methods which allow sampling from such densities—the class of *Markov chain Monte Carlo (MCMC)* methods [22]. MCMC methods are well-known in the context of Bayesian inference; however, for completeness a summary of the method used here—the *Metropolis-Hasting* (MH) algorithm—is provided here.

It is sufficient for now to assume that one has a single random variable x with true, but unknown, density $p(x)$. The MH method actually works even if $p(x)$ is too complicated to sample from directly. First of all, one assumes that one can at least *evaluate* $p(x)$, given a candidate value for x; in fact, one only needs to evaluate a density $p^*(x)$, where $p^*(x) = Z_p p(x)$, i.e. the density of interest up to a multiplicative constant. The method hinges on the use of a *proposal distribution* $q(x)$, which is simpler than $p(x)$ and can also be evaluated up a multiplicative constant. The use of the proposal distribution is common to many methods like importance and rejection sampling [22]. The unique feature of MCMC methods is that a sequence of samples or states x^i are generated, with the proposal distribution a function of the state at any given time. One estimates the probability for the next state in the sequence (a *Markov chain*—hence the name) by conditioning on the current state, the proposal distribution is $q(x'|x^i)$. Another way of regarding q is as a transition probability density for a jump in the state/sample space. Often q is taken as a simple distribution, i.e. a Gaussian. Under the latter assumption, one can see that a large variance for the proposal distribution will lead to higher probabilities of jumps in the state/sample space. One also immediately sees that a small variance, leading to small jumps in the space, will produce (at least locally) correlated states/samples.

The MH algorithm proceeds as follows. Assume one is at iteration i in the process:

1. Sample from the proposal density $q(x'|x^i)$ to generate a candidate state x'.
2. Evaluate the quantity:

$$a = \frac{p^*(x^i)q(x^i|x')}{p^*(x')q(x'|x^i)}$$

3. If $a \geq 1$, the new state is accepted; *otherwise* the new state is accepted with probability a.
4. If the new state is accepted, set $x^{i+1} = x'$; else set $x^{i+1} = x^i$.

As a full description of why this procedure leads to sampling from $p(x)$ is beyond the scope of this tutorial, the curious reader should consult [22].

Two practical issues arise with the iteration above. The first concerns initial conditions. As in any iterative process, initial conditions matter; depending on these there will be a transient period before the sequence becomes stationary and is consistently generating samples from $p(x)$. In order to allow for this, the algorithm is usually run for a *burn-in* period before samples are drawn. The second issue is concerned with independence; as observed above, the proposal density may only allow small jumps in the state/sample space, so states close together in the sequence will be *correlated*. To obtain independent samples, one only saves every n_ith state; this process is called *thinning*. Ideally, one chooses the thinning interval long enough for the states to potentially travel across the whole support of the density.

The ability to sample from a distribution provides access to a great deal of information. Suppose one is interested in a function F of the random variables of interest $\underline{x}$; the main quantity of interest is likely to be the expectation:

$$E[F(\underline{x})] = \int F(\underline{x})p(\underline{x})d\underline{x} \tag{8.12}$$

The basis of the Monte Carlo (MC) method is that, if one can draw a sequence of samples $\{\underline{x}_1, \ldots, \underline{x}_N\}$, from $p(\underline{x})$, one has the approximation:

$$E[F(\underline{x})] = \int F(\underline{x})p(\underline{x})d\underline{x} \approx \frac{1}{N}\sum_{i=1}^{N} F(\underline{x}_i) \tag{8.13}$$

which has the critical property that the accuracy of the estimates *does not depend on the dimension of* $\underline{x}$ [22].

Furthermore, if samples from the distribution are available, one can easily construct parametric and nonparametric (e.g. kernel density) estimates of the distribution.

Given that MCMC, in particular the MH method, allows one to draw samples from a given distribution and earlier discussions explained how this can lead to system identification via estimation of parameter distributions and statistics, what remains to be explained is the specific detail of how this is implemented here.

In order to implement MCMC, it is necessary to evaluate the density $p^*(\underline{w}|\mathcal{D})$, which is the required $p(\underline{w}|\mathcal{D})$ up to a constant multiple. Using Bayes theorem, one has:

$$p(\underline{w}|\mathcal{D}) = \frac{p(\mathcal{D}|\underline{w})p(\underline{w})}{p(\mathcal{D})} \tag{8.14}$$

and as $p(\mathcal{D})$ is fixed by the data (and cannot be evaluated anyway), one has:

$$p^*(\underline{w}|\mathcal{D}) = p(\mathcal{D}|\underline{w})p(\underline{w}) \tag{8.15}$$

as the required density (p, up to a constant). This object is the product of a likelihood for the data (which can be computed under the assumption that the noise is Gaussian—the log likelihood is then related to the least-squared error between predictions using the candidate parameters and the measured data) and a prior for the parameters. If very little is known about the system, an uninformative (constant, and therefore improper) prior can be assumed, and then one can take:

$$p^*(\underline{w}|\mathcal{D}) = p(\mathcal{D}|\underline{w}) \tag{8.16}$$

The only thing needed now for the MH algorithm is the proposal distribution $q(\underline{x}'|\underline{x}^i)$, and arguably the simplest prescription for this is a spherical Gaussian distribution:

$$q(\underline{x}'|\underline{x}^i) \sim N(\underline{x}^i, \sigma_p I) \tag{8.17}$$

controlled by a single scale parameter σ_p (I is the identity matrix).

None of the above discussion is specific to differential equation models; in fact the algorithm is applicable to any type of predictive model. Differential equations are singled out here because they often represent systems in structural dynamic problems.

8.4 Case Study in Parameter Estimation: The Duffing Oscillator

The training data for the current study were generated by simulation. The equation of motion considered will be the classical Duffing oscillator of Eq. (8.2). The data were generated here using Matlab [23], using a fixed-step fourth-order Runge-Kutta scheme for initial value problems [24]. The parameters for the baseline system here were: $m = 1$, $c = 20$, $k = 1.0 \times 10^4$ and $k_3 = 5 \times 10^9$. The excitation $x(t)$ was initially a Gaussian random sequence with mean zero and standard deviation 10.0. The step size (or sampling interval) was taken as 0.001 s, corresponding to a sampling frequency of 1000 Hz. Noise of RMS 1% of the response was added to the displacement signal which was used, with the corresponding samples of excitation force, as 'training data' for the algorithm. The training set used here was composed of 500 points corresponding to a record duration of 0.5 s. The clean training set, force and displacement (before the addition of the noise) is shown in Fig. 8.1.

Once the data had been generated, the MCMC algorithm discussed in the last section was applied to the identification problem. The MCMC identification routine was coded in the Python language using the package PyMC [25]. As well as the four parameters: m, c, k and k_3, the MCMC routine also learned the output noise variance; however, for convenience this was encoded as a precision parameter $\beta = 1/\sigma_\epsilon^2$. Because of the differences in scale between the parameters, log values were used in the algorithm to improve conditioning. All of the model parameters were given uniform priors spanning a range one order of magnitude above and below the true parameter values; the initial conditions for the Markov chain were set close to the true parameter values. The proposal distribution q was set as a spherical Gaussian with variance of 0.1 in each direction. The MCMC algorithm was set to run for 25,000 samples, thinning by taking every 10th sample, and a burn-in of 5000 samples was used. The results from the MCMC run, summarising 2000 samples altogether, are shown in Fig. 8.2. Each of the subfigures shows the sequence of MC samples for the parameter of interest, followed by a frequency histogram giving a coarse view of the parameter density.

The first observation one can make from Fig. 8.2 is that the chains for the parameters m, c and β are largely stationary immediately following the burn-in period, while the parameters k and k_3 may still be undergoing some longer scale fluctuations. One also observes that the long-term behaviours of the latter two parameters are anticorrelated. This is understandable in that some of the true cubic behaviour can be accommodated by biasing the linear term. There is also (less evident) longer-term variation in the m parameter which is correlated with the k parameter. This is also understandable;

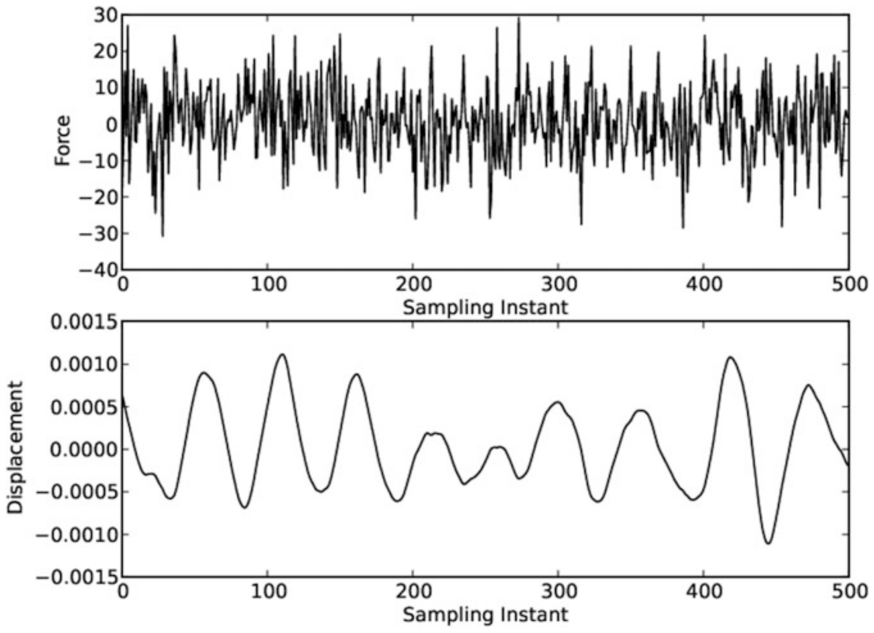

Fig. 8.1 Training data for Duffing system case study

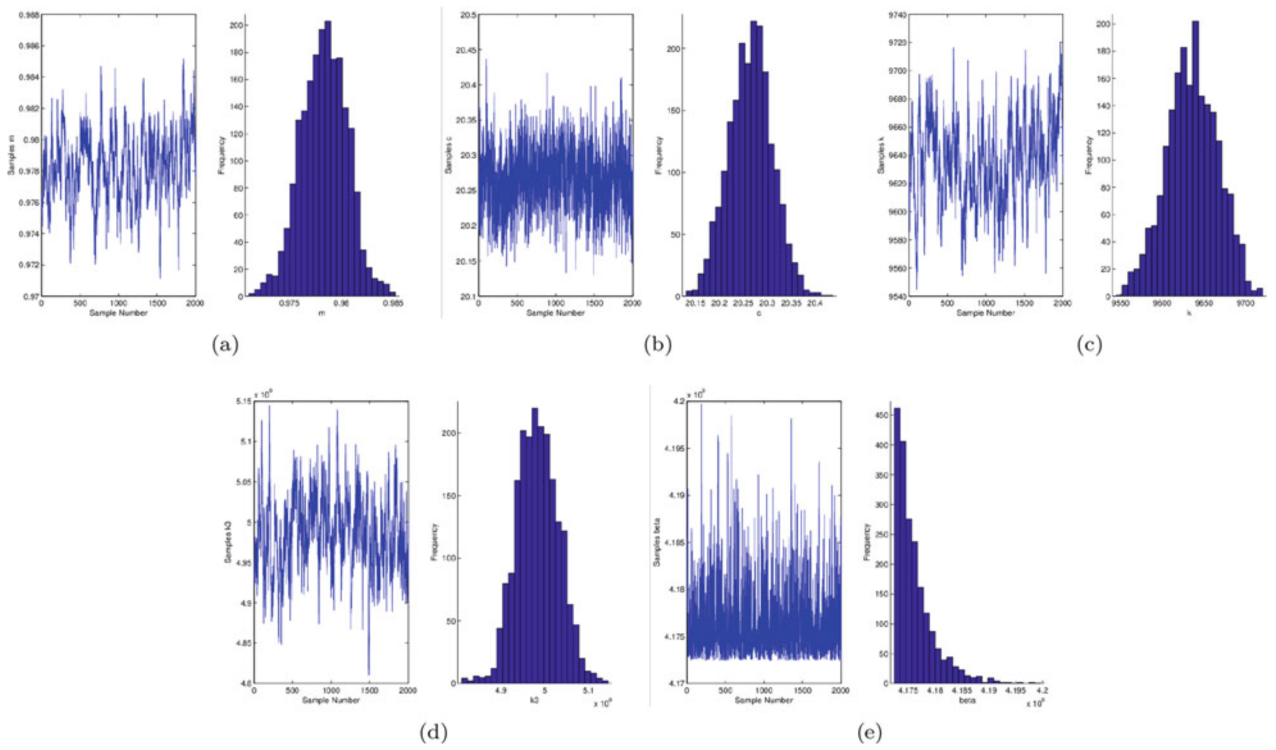

Fig. 8.2 Results from MCMC identification of Duffing oscillator system. (**a**) m. (**b**) c. (**c**) k. (**d**) k_3. (**e**) β

Table 8.1 Parameter estimates for Duffing oscillator model

Parameter	True value	MC mean	MC std
m	1.0	0.978	0.002
c	20.0	20.265	0.045
k	1.0×10^4	9635.7	30.7
k_3	5.0×10^9	4.983×10^9	0.048×10^9

in order to explain the data, the model needs to have the correct natural frequency $\omega_n = \sqrt{k/m}$, and any fluctuations in k should be mirrored in m in order to keep the correct ω_n for the system. When the mean and standard deviations for the parameters were estimated over the samples in each chain, the results were as summarised in Table 8.1.

The results show a little bias in the sense that the true parameters are not within the 3σ bounds of the estimates. This bias is because the chains are not yet stationary by the end of the calculation, partly because of the long-run correlations between the parameters.

As an objective measure of fidelity of the model predictions, one can define a *normalised mean-square error* by:

$$NMSE(y^*) = \frac{100}{N\sigma_y^2}\sum_{i=1}^{N}(y_i - \backslash ys_i)^2 \tag{8.18}$$

where the y_i are the observed outputs and the y_i^* are the predictions. Experience with this measure has shown that a value of less than 5.0 generally indicates a good model, while a value less than 1.0 suggests excellence. With the MCMC parameters, the NMSE for the predictions here is 0.53, which indicates an excellent fit to the data. In contrast to an identification scheme where the parameter distributions are *assumed* Gaussian, the parameter samples are available here so that a test can be carried out to check. Also nonparametric estimators for the densities can be applied. Using the last 50 samples from the chains, the predicted responses for the training inputs were computed and are shown superimposed on the true response in Fig. 8.3. There is very little variation between the realisations; this requires further discussion.

The low degree of variation between realisations is a consequence of the way the predictions have been made and is on the optimistic side. By integrating the differential equations across the entire time with a single realisation of the parameters, the predictions ignore the component of uncertainty associated with *state estimation*. One might argue that the correct course

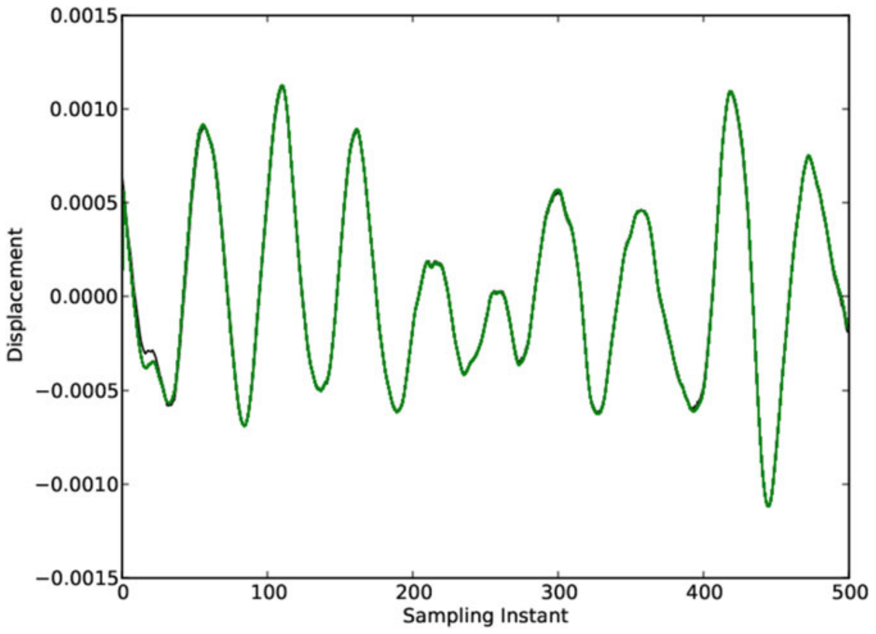

Fig. 8.3 Predictions using parameters from MCMC parameter samples, superimposed on observed data

Table 8.2 Parameter estimates for Duffing oscillator model: high noise case

Parameter	True value	MC mean	MC std
m	1.0	0.982	0.01
c	20.0	20.40	0.13
k	1.0×10^4	9693.3	81.5
k_3	5.0×10^9	4.99×10^9	0.13×10^9

of action would be to integrate forward one step at a time, sampling from the parameter distribution, to make an estimate of the response at the next time step. This strategy will clearly increase the overall bounds on the predictions. In order to expose the effect under discussion more clearly, a further identification of the Duffing data was carried out, this time adding noise of standard deviation of 5% of the signal before the MCMC process. The training data is as before for the system, but for this example an independent test set of data was used to assess the predictive capability of the model. The parameter estimates in this case are summarised in Table 8.2.

As one would expect, the parameter estimates in the higher noise case have greater standard deviations as compared to those in Table 8.1. This time, in order to show the effects of state estimation, the predictions were repeated but this time sampling from the MCMC parameter samples at *each time step* in the Runge-Kutta process. 50 realisations were made and the mean prediction was extracted as was the variance in the predictions. In order to accommodate the fact that the 'measured' data here are noisy, one has to add the MCMC-estimated noise variance for the computed bounds. The result of this calculation is shown in Fig. 8.4, which now has the measured data falling within the prediction intervals.

8.5 Model Selection and Approximate Bayesian Computation

The problem of *structure detection* or *model selection* discussed in the introduction here is extremely difficult when approached in its full generality. Unlike parameter estimation which searches some finite-dimensional parameter space to find a minimum according to some error criterion, structure detection requires searching some infinite-dimensional function space. In the simplest case, one imagines replacing the y^3 in Duffing Eq. (8.2), with some arbitrary stiffness function $f(y)$; even with constraints like continuity or differentiability, the search space is infinite-dimensional. Leaving aside intermediate possibilities for the moment, the most severe constraint one could impose is to propose a finite set of *candidate functions*, from which to choose. An offline approach to the identification would then be to run the MCMC algorithm for each candidate model and then select the best according to some validation criterion. Although this would work, it involves multiple identification runs and would potentially be computationally expensive; it also lacks elegance. A more elegant solution

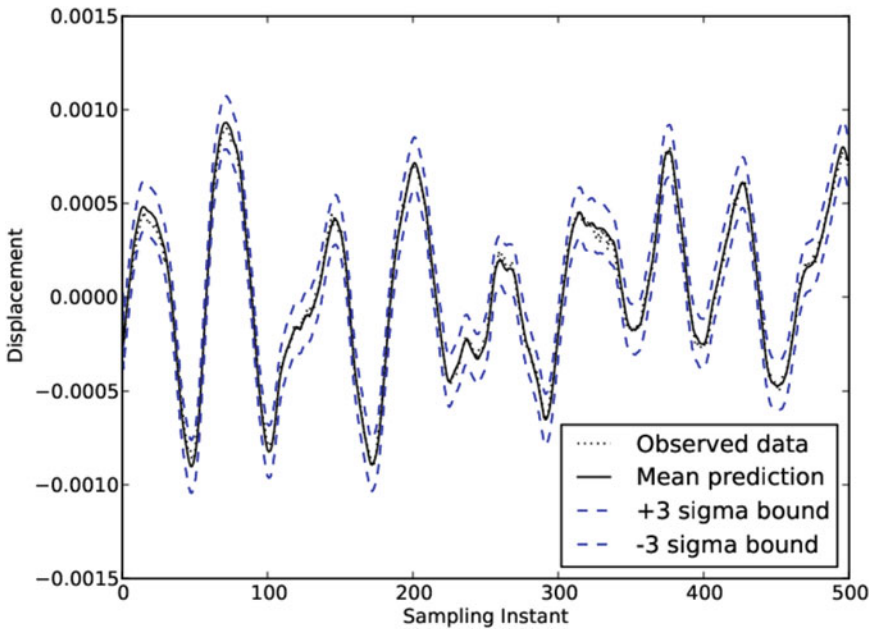

Fig. 8.4 Predictions using parameters from MCMC parameter samples identified in higher noise case. Bounds take into account parameter and state uncertainty and noise variance

would carry out both structure detection and parameter identification within one identification run. Such an algorithm would ideally trade off model complexity against model fidelity and thereby protect against the possibility of *overfitting*. Such algorithms which carry out both model selection and parameter estimation have recently been grouped under the term *equation discovery*. As Bayesian algorithms immediately suggest themselves, they implicitly implement such a trade-off [22].

What then would such a model look like? Having just seen the power of MCMC for parameter estimation, one might conceive of a similar algorithm where the Markov chain not only jumps between values in the parameter space but also jumps between models in the candidate set. Such methods do exist, but they are necessarily more complex than the MH algorithm discussed earlier. One of the problems can be seen immediately by considering the situation where the candidate set is {linear, Duffing}, where the linear model structure $\mathcal{M}_1$ is given by Eq. (8.1) and the Duffing structure $\mathcal{M}_2$ is given by (8.2). In this case, the parameter space for $\mathcal{M}_1$ is three-dimensional, while that for $\mathcal{M}_2$ is four-dimensional; the Markov chain would need to move between spaces of different dimension. Designing an MCMC algorithm with such a capability turns out to be quite difficult if one wishes for the convergence guarantees that hold for MH, etc. [22]. One algorithm with the desired properties is the *reversible-jump MCMC* algorithm [26]. While RJ-MCMC has been applied for NLSI [27, 28], it is difficult to code, is computationally expensive and depends on a number of algorithm parameters (hyperparameters), which require careful tuning. Fortunately, there is a 'simpler' alternative—*approximate Bayesian computing* (ABC).

The ABC algorithm has been applied in many contexts for equation discovery: genetics [29], biology [30, 31] and psychology [32]. It soon attracted attention as a promising method in NLSI [15, 33]. There are a number of variations on the ABC algorithm; for example, the approach in [33] was based on subset selection, while that of [15] was based on sequential MC (SMC). The SMC variant of the algorithm will be described and applied in the following. ABC offers the possibility of managing larger datasets and higher numbers of competing models with different dimensionalities, circumventing the limitations of RJ-MCMC.

In the core ABC algorithm, the objective is to obtain a good and computationally affordable approximation to the posterior distribution:

$$p(\underline{w}|\mathcal{D}^*, \mathcal{M}) \propto p(\mathcal{D}^*|\underline{w}, \mathcal{M})p(\underline{w}|\mathcal{M}) \tag{8.19}$$

where $\mathcal{M}$ is the model controlled by the set of parameters $\underline{w}$, $p(\underline{w}|\mathcal{M})$ denotes the prior distribution over the parameter space and $p(\mathcal{D}^*|\underline{w}, \mathcal{M})$ is the likelihood of the observed data $\mathcal{D}^*$ for a given parameter vector $\underline{w}$.

ABC was originally designed as a *likelihood-free* method to overcome issues from the intractable likelihood functions encountered in various real-world problems; as such, it relies on systematic comparisons between *observed* and *simulated* data. The main principle consists of comparing the simulated data, $\mathcal{D}$, with observed data $\mathcal{D}^*$, and accepting simulations if a suitable distance measure between them, $\Delta(\mathcal{D}, \mathcal{D}^*)$, is less than a specified threshold ε, defined by the user. The ABC algorithm thus provides a sample from the approximate posterior of the form:

$$p(\underline{w}|\mathcal{D}^*, \mathcal{M}) \approx p_\varepsilon(\underline{w}|\mathcal{D}^*, \mathcal{M}) \propto \int f(\mathcal{D}^*|\underline{w}, \mathcal{M}) \mathbf{1}\left(\Delta(\mathcal{D}, \mathcal{D}^*) \leq \varepsilon\right) p(\underline{w}|\mathcal{M}) \mathrm{d}\mathcal{D} \tag{8.20}$$

where $\mathbf{1}\left(\Delta(\mathcal{D}, \mathcal{D}^*) \leq \varepsilon\right)$ is an indicator function which is unity if the bracketed condition is met and zero otherwise. When ε is small enough, $p_\varepsilon(\underline{w}|\mathcal{D}^*, \mathcal{M})$ will be a good approximation to the true posterior distribution.

In this work, the ABC-SMC algorithm of [30] will be illustrated. Generally speaking, the algorithm works as a *particle filter* which can be used to identify nonlinear dynamical systems [34]. The algorithm is based on the *sequential importance sampling* (SIS) algorithm, the Monte Carlo (MC) method which has formed the basis for most (SMC) algorithms developed in the past [35]. The key idea of ABC-SMC is to represent the required posterior density function by a set of random samples (*particles*) with associated weights. The algorithm iterates via a number of intermediate posterior distributions, before converging to the optimal approximate posterior distribution satisfying a convergence criterion defined by the user.

Algorithm 1 ABC-SMC for model selection

Input: Observed data u^*, n competing models $\mathcal{M}^n_{k=1}$, tolerance threshold ε_1, prior distributions $\pi(\alpha)$, $\pi(\mathcal{M}_k)$
Output: Model posterior probabilities, parameter distributions

```
1:  At iteration t = 1,
2:  for $i = 1 : N$ do
3:      repeat
4:          Select $\mathcal{M}^* = m_k$ from the prior distribution.
5:          Select $\alpha^*_k$ from the prior: $\pi(\alpha_k|m_k)$.
6:          Simulate $u$ from $\mathcal{M}_k(u|\alpha^*_k)$.
7:      until $\Delta(u^*, u) < \varepsilon_1$.
8:      Set the particle as $\mathcal{M}^{(i)}_1 = m_k$ and $\alpha^{(i)}_{\mathcal{M}_1} = \alpha^*_k$ with weight $\omega^{(i)}_1 = \frac{1}{N}$.
9:  end for
10: for $t = 2, \ldots, T$ do
11:     for $i = 1, \ldots, N$ do
12:         repeat
13:             Select $\mathcal{M}^* = m_k$ from the prior distribution.
14:             Sample $\alpha^{(i)}_{k,t-1}$ with corresponding weights $\omega^{(j)}_{t-1}$ and perturb the particle by generating $\alpha^*_k$.
15:             Simulate $u$ from $\mathcal{M}(u|\alpha^*)$.
16:         until $\pi(\alpha^*_k|m_k) > 0$ and $\Delta(u, u^*_k) < \varepsilon_k$
17:         Set the particle as $\mathcal{M}^{(i)}_t = m_k$ and $\alpha^{(i)}_{\mathcal{M}_t} = \alpha^*_k$ with weight:
18:             $\omega^{(i)}_t = \frac{\pi(\alpha^*_k|m_k)}{\sum_{j=1}^{N} \omega^{(j)}_{t-1} K_\alpha\left(\alpha^*_k|\alpha^{(j)}_{k,t-1}\right)}$
        end for
19:     For every $m_k$, $k = 1, \ldots, \ell$, normalise the weights.
20: end for
```

Following the scheme shown in Algorithm 1, one starts the iteration with an arbitrarily large tolerance threshold ε_1 to avoid a low acceptance rate and thus computational inefficacy. One selects directly from the prior distributions $p(\mathcal{M})$ and $p(\underline{w})$, evaluates the distance $\Delta(\mathcal{D}^*, \mathcal{D})$ and then compares this distance to ε_1, in order to accept or reject the $(\mathcal{M}, \underline{w})$ selection. This process is repeated until N particles distributed over the competing models are accepted. One then assigns equal weights to the accepted particles for each model. For the next iterations ($t > 1$) the tolerance thresholds are set such that $\varepsilon_1 > \varepsilon_2 > \ldots > \varepsilon_t$. The choice of the final tolerance schedule denoted here by ε_t depends mainly on the goals of the practitioner.

The dynamics by which ε evolves is a matter of choice; there is no general prescription. The tolerance threshold can be selected manually or adaptively, based on the distribution of the accepted distances in the previous iteration, $t - 1$. For example, the threshold of the second iteration can be set to the p^{th} percentile of the distances in the first iteration. This would seem to be the most common choice of tolerance threshold sequence since it is intuitive and simple to define. Both methods were used in this work and seem to do very well, in the sense that an appropriate acceptance rate is maintained over the populations. For the second strategy, it was found that a percentile between 20 and 40 is a rational choice. Once ε_2 is set, one selects a model and a particle from the previous weighted set of particles and perturbs it via a predefined kernel; again the selection of the kernel is a matter of choice. A widely used option is to define the perturbation kernel as a multivariate Gaussian centred on the mean of the particle population, with a covariance matrix set to the covariance of the population obtained in the previous iteration. For a deep discussion of various schemes for specifying the perturbation kernels, the reader is referred to [36].

In this study, the particle perturbation distribution is uniform and symmetric around zero, with the interval width (for each parameter) taken as the range of the parameter in the previous population. One then calculates the distance $\Delta(\mathcal{D}^*, \mathcal{D})$, compares to the new tolerance threshold and accepts the new particle if $\Delta(\mathcal{D}^*, \mathcal{D}) \leq \varepsilon_2$; otherwise the particle is rejected. This process is repeated until a new set of N particles is assembled. One then updates the particle's weight according to the kernel. The entire procedure is repeated until convergence is met. One way to accept convergence is to impose a target threshold close to zero; another is to control the acceptance ratio, which is measured at each iteration. This ratio is the quotient of the number of proposals to the full number of proposed particles at every step. When this ratio falls below some given limit, the algorithm is halted. There are actually numerous meaningful ways to establish convergence [15].

When the algorithm halts, the approximate marginal posterior distribution for model candidate $\mathcal{M}_i$ is estimated by:

$$P(\mathcal{M}_i|\mathcal{D}^*) \approx \frac{\text{Accepted particles for } \mathcal{M}_i}{\text{Total number of particles } N} \tag{8.21}$$

As one can see, the algorithm does require the selection of a number of hyperparameters—the sequence of acceptance thresholds. A careful choice of those hyperparameters is important, since they will affect performance of the algorithm. A bad choice may lead to computational expense and/or biased parameter estimates.

8.6 Case Study in Equation Discovery: Hysteretic Systems

Hysteretic systems—or systems with *memory*—have always presented problems in terms of system modelling. The models often contain terms which are highly (and/or discontinuously) nonlinear in the parameters, and they may also evolve via unmeasured states. Even parameter estimation can be challenging. One compact and versatile model class which is popular for hysteretic SI is the *Bouc-Wen* (BW) class [37, 38].

The general single-degree-of-freedom (SDOF) hysteretic system described in the terms of Wen [38] is represented below, where $g(y, \dot{y})$ is the polynomial part of the internal restoring force, $z(y, \dot{y})$ is the hysteretic part and $x(t)$ is the excitation, as usual:

$$m\ddot{y} + g(y, \dot{y}) + z(y, \dot{y}) = x(t) \tag{8.22}$$

where m is the system mass. In the following discussion, the polynomial part of the internal force will be considered linear: i.e. $g(y, \dot{y}) = c\dot{y} + ky$.

The hysteretic component is defined by Wen [38], via the additional equation of motion:

$$\dot{z} = \begin{cases} -\alpha|\dot{y}|z^n - \beta\dot{y}|z^n| + A\dot{y}, & \text{for } n \text{ odd} \\ -\alpha|\dot{y}|z^{n-1}|z| - \beta\dot{y}|z^n| + A\dot{y}, & \text{for } n \text{ even} \end{cases} \tag{8.23}$$

The parameters α, β and n govern the shape and the smoothness of the hysteretic loop. The equations offer a simplification from the point of view of parameter estimation, in that the linear stiffness term from g in Eq. (8.22) can be combined with the $A\dot{y}$ term in the state equation for z. The reader can refer to [14] for full details.

In this illustration, the response output will be assumed to be displacement. Data were simulated in the same manner as for the Duffing example earlier. The sampling interval was taken as 0.001 s, corresponding to a sampling frequency of 1000 Hz. Here, as in the first example, the excitation is Gaussian with zero mean, but with a standard deviation of 10. The exact parameter values used to generate the training data and the parameter ranges are summarised in Table 8.3. Figure 8.5 shows the BW model response with $n = 2$. The training data used here were composed of 1001 points, corresponding to a record duration of 1 s.

Table 8.3 Parameter ranges of the Bouc-Wen model

Parameter	True value	Lower bound	Upper bound
m	1	0.1	10
c	20	2	200
α	1.5	0.15	15
β	−1.5	−15	−0.15
A	6680	668	66,800

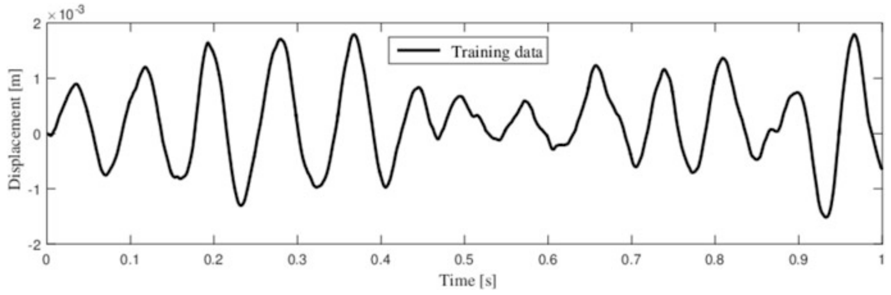

Fig. 8.5 Training data

The equation discovery problem here is created by proposing a range of potential models: a linear model and four BW models with different values of n.[5] Thus, five competing models are considered, denoted by,

$$\mathcal{M}_1 \quad : \quad m\ddot{y} + c\dot{y} + Ay = x(t) \tag{8.24}$$

$$\mathcal{M}_{2:5} \quad : \quad \text{Eqs. (8.22) and (8.23)}, \;\; n = 1:4 \tag{8.25}$$

To implement the ABC-SMC algorithm here, equal prior probabilities $p(\mathcal{M}_{i=1:5}) = \frac{1}{5}$ were assumed. In this example, the tolerance threshold sequence is adaptively defined. It was found that a tolerance threshold set at 20th percentile of the particle distances from the previous iteration maintained a quite satisfactory acceptance rate through the algorithm run. The number of particles in the population was taken as 1000, and the distance metric simply used the NMSE defined in Eq. (8.18) earlier. The algorithm stopped when the difference between two consecutive tolerance thresholds was less or equal to 5×10^{-4}. The results obtained are summarised below (using noisy measurements, 1% RMS was added to the response).

Figure 8.6 shows how the ABC-SMC algorithm progressively eliminated the least likely models. One observes that $\mathcal{M}_5$ was the first to be eliminated which means that it is least adequate to explain the data. The linear model is removed in the population after. The models $\mathcal{M}_{2\to4}$ appear to be the most likely candidates to explain the data, since after eliminating $\mathcal{M}_5$ and $\mathcal{M}_1$, it seems quite difficult to favour one model above the other. Only at Population 18 ($\varepsilon = 0.0058$) does the algorithm start to favour $\mathcal{M}_3$; this would suggest that, at this level of error, even $\mathcal{M}_2$ and $\mathcal{M}_4$ may explain the data reasonably well, with preference for $\mathcal{M}_3$. The algorithm converges on the true model at Population 19 ($\varepsilon = 0.0042$). Figure 8.7 shows the histograms of the model parameter values, derived from the final population; the parameter statistics are presented in Table 8.4. Apart from α, which is outside the (5th, 95th) percentiles, the other parameters are well estimated. The training data and the model prediction are shown in Fig. 8.8, from which one can see the accuracy of the model.

The ABC-SMC algorithm is shown to be efficient in problems with several competing models; significantly, those models are quite similar.

Of course, there are many other approaches to equation discovery; the subject is currently experiencing a great deal of interest. One class of algorithms which is currently popular is based on the idea of a linear expansion over a dictionary. The idea is quite simple in principle; the equation required is proposed to be of the form (consistent with applications in structural dynamics):

$$\sum_{i=1}^{N} w_i f_i(\ddot{y}, \dot{y}, y, x) = 0 \tag{8.26}$$

[5] Of course, one could simply consider n as one of the parameters to be estimated, but this creates problems of its own.

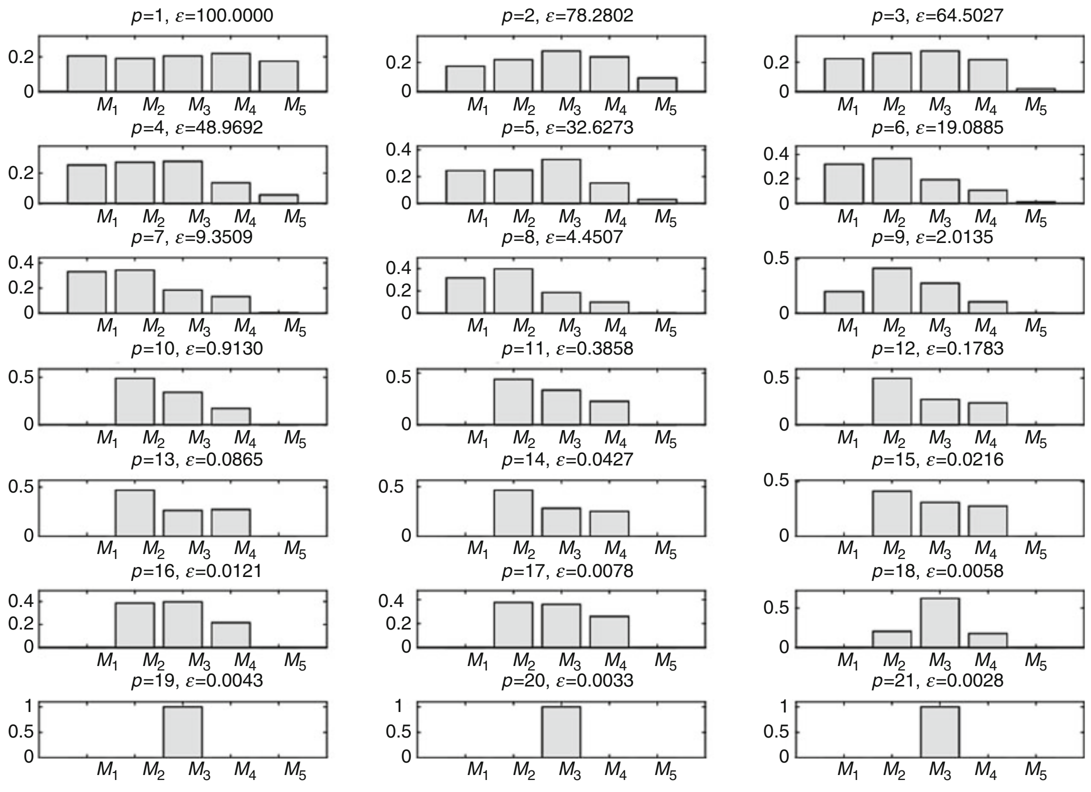

Fig. 8.6 Model posterior probabilities of the Bouc-Wen model over the populations

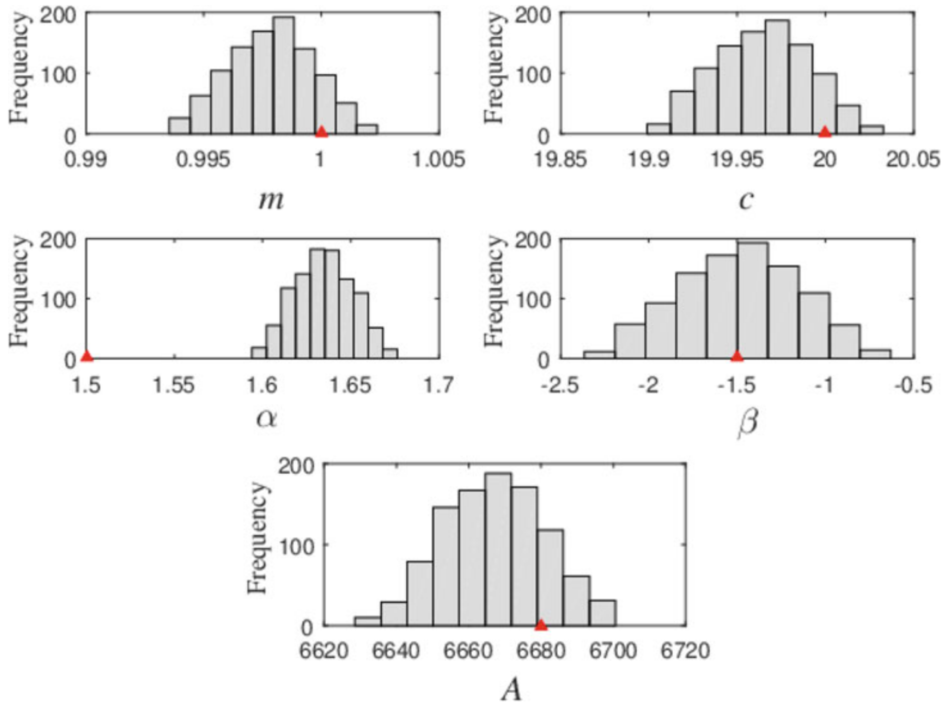

Fig. 8.7 Histograms of the identified BW model parameters (the red triangles show the true values)

where the w_i are tunable parameters and the f_i are N functions of the variables of interest (and their derivatives), *without* tunable parameters. The f_i are selected from a potentially large set of candidate terms—the *directionary*. In principle, there is no demand that the f_i form a complete set of basis functions. To give a concrete example, the dictionary $\{\ddot{y}, \dot{y}, y, y^3, x\}$ would

Table 8.4 Model parameter estimates from the last population

Parameter	True value	Mean, μ	Std. dev, σ	[5th, 95th] percentiles
m	1	0.9976	0.0017	[0.9947, 1.0004]
c	20	19.965	0.0248	[19.9241, 20.006]
α	1.5	1.6339	0.0149	[1.6083, 1.6577]
β	-1.5	-1.5005	0.3124	[-2.031, -0.9937]
A	6680	6665.1173	13.148	[6643.3231, 6687.1163]

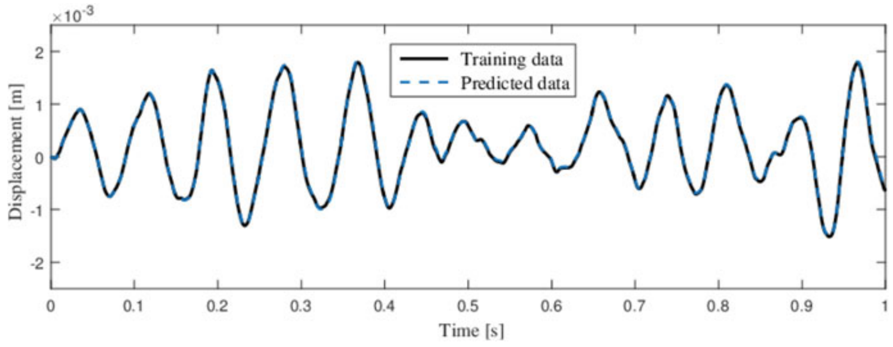

Fig. 8.8 Comparison between the training and predicted data

be sufficient to identify both the linear system in (8.1) and the Duffing oscillator in (8.2). In general, the dictionary would have very many terms, to increase the probability of discovering the correct equation; the issue then becomes the classic problem of overfitting. The solution is to find some means of enforcing sparsity in the model, i.e. driving the algorithm to select as few terms as possible in the model, by forcing as many parameters w_i as possible to zero [39–41]. One simple *shrinkage* approach is to monitor the contributions of individual terms to the overall variance and then zero any terms with contributions below some small threshold. There are many elegant Bayesian approaches to the problem, including using *automatic relevance determination* (ARD) priors over the parameters [2, 42]. More recent developments include the use of technology like *spike-and-slab* priors [43, 44].

8.7 Coloured Boxes and Physics-Informed Machine Learning

Here they come, every colour of the rainbow: black, white, brown.[6]

This section will return to the discussion of white, grey and black boxes; most of the material here follows the work of the second author during a recent fellowship. Before launching into a discussion of the technology, it is useful to revisit the basic terminology. In the introduction, a model which was determined *completely* by physics was termed a *white-box* model; in fact, it will be useful to relax this definition somewhat. While true white-box models do crop up in physics, they are comparatively rare in engineering. This fact occurs because even the most well-constructed systems and structures will deviate from their designs in some way, because of the imperfect nature of fabrication. In reality, one may know the *structure* of system equations of motion, but the exact parameters in operation will deviate from the design. For this reason, a white-box model here will be redesignated as one in which the model structure is not completely determined by the physics. If one needs to refer to the original concept, one could use the term *snow-white* model. With this amendment, a *grey-box* model is one for which the model structure and any parameters are undetermined by physics alone.

As stated in the introduction, it is generally unwise to construct pitch-black models using machine learners uninformed at all by physics. In some cases, this course of action is forced upon the modeller, but in general, some physical or engineering insight will be present and it is simply wasteful to ignore it. There are potentially many ways to construct a grey box model, but the simplest in principle is to form a direct sum of a white box and a black box, as in:

$$y(t) = F_t[\tilde{y}, \tilde{x}] + \Delta_t[\tilde{y}, \tilde{x}] + \varepsilon(t) \tag{8.27}$$

[6] Anonymous British snooker commentator.

where F_t is the white component and Δ_t is the black;[7] $\varepsilon(t)$ is assumed to be a noise sequence. In general F_t will be a structure suggested by physics with undetermined parameters with some physical significance; in contrast, Δ_t will be a nonparametric learner with parameters (and hyperparameters), devoid of physical significance.

Even this *basic* grey box presents technical problems if F_t and Δ_t are to be fitted simultaneously. One simple alternative strategy is to fit the white-box model, assuming that $y(t) = F_t[\tilde{y}, \tilde{x}]$, to then form the *residual* $r(t) = y(t) - F_t[\tilde{y}, \tilde{x}]$, and finally to fit $r(t) = \Delta_t[\tilde{y}, \tilde{x}]$. With this approach, the white-box model is primary, and the black box is considered a *residual model*, mopping up any systematic behaviour in y that is unaccounted for in F_t. Philosophically, Δ_t is countering any epistemic uncertainty in the physics and can be regarded as a *model discrepancy* term [45]. In the much simpler situation where y is a direct static function of x, the decomposition becomes:

$$y = \underbrace{f(x)}_{White-box} + \underbrace{\delta(x) + \varepsilon}_{Black-box} \tag{8.28}$$

A nice example of the application of a residual model can be found in [46], which is concerned with predicting wave-loading forces on offshore structure. For many years, the industry standard in predicting wave forces has been *Morison's equation*. However, Morison's equation is well-known to predict badly under certain Circumstances, e.g. when vortex shedding is present in the fluid kinematics. The approach in [46] adopts Morison's equation as the white-box component of the model:

$$y_t = \underbrace{C_d' U_t |U_t| + C_m' \dot{U}_t}_{Morison's\ Equation} + \underbrace{f([u_t,\ u_{t-1}, \dots,\ u_{t-l_u},\ y_{t-1},\ y_{t-2}, \dots,\ y_{t-l_y}]) + \varepsilon}_{GP-NARX} \tag{8.29}$$

where y_t is the wave force, C_d' is the *drag coefficient*, C_m' is the *inertia coefficient*, U is the wave velocity and $\dot{U}$ is the wave acceleration. In this case, the black-box component is a Gaussian process nonparametric learner, made dynamic by imposing a NARX structure which regresses the output on past inputs and outputs—a GP-NARX model [47]. In the GP-NARX component, $u_{t:t-l_u}$ are lagged exogenous inputs and $y_{t-1:t-l_u}$ are the lagged wave force; see [46] for more details. The grey box gives significant improvements on Morison's equation alone.

The term 'grey-box model' is perhaps most familiar to those from a control engineering background. Sohlberg [48, 49] provides a useful review and overview of grey-box models in this context. The term 'hybrid model' is sometimes used as a synonym for a grey box. Figure 8.9 attempts to capture and summarise some of the currently available modelling approaches relevant for challenges in structural dynamics on the white to black spectrum. Note that the "degree of greyness" of the models in the middle region will change according to implementation and application.

At the whiter end of the spectrum are modelling approaches where data are used for parameter estimation or model form selection (with equation discovery fitting in here). Considering the spectrum beyond residual models, the term *hybrid architectures* reflects the wider possibilities for combinations of white and black models (which could include the summation forms of (8.27) and (8.28)). The remainder of the spectrum contains models with structures that are more data-driven/black-box in nature. Sohlberg [49]) describes *semi-physical modelling* as when features are subject to a nonlinear transformation before being used as inputs to a black-box model; this is also referred to as *input augmentation*, see [50, 51] for more examples. These examples are placed under the heading of 'manipulation of black-box inputs'. *Constraints for machine learning algorithms* refers to methods that allow one to constrain the predictions of a machine learner so that they comply with physical assumptions. The final grouping of grey-box approaches mentioned here is *physics-guided black-box learners*. These are methods that use physical insight to attempt to improve model optimisation and include the construction of physics-guided loss functions and the use of physics-guided initialisation; these will not be discussed further here, but see, e.g. [52] for more details. The whole spectrum (with the exception of the pitch black) delineates the modern field of *physics-informed machine learning*.

The next illustration will concern *constraints for machine learning algorithms*; the machine learning approach here will be *Gaussian process* (GP) regression. GPs have been shown to be a powerful tool for regression tasks [53], which are becoming common in structural dynamic applications. Their use here and throughout the work of the authors is because of their (semi)non-parametric nature, their ability to function with a small number of training points and most importantly, the Bayesian framework within which they naturally work. The Gaussian process formulation provides a predictive *distribution*

[7] The notation here is intended to be flexible enough that it can represent many of the standard mathematical structures. The tilde is used to suggest that information for all times is available for the x and y. F_t and Δ_t are then *functionals* which can make use of any variable information *up to* time t. In this sense the notation is purely formal.

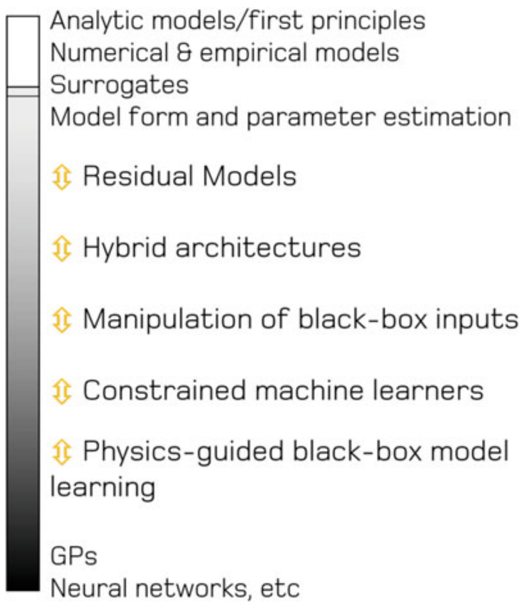
Analytic models/first principles
Numerical & empirical models
Surrogates
Model form and parameter estimation
Residual Models
Hybrid architectures
Manipulation of black-box inputs
Constrained machine learners
Physics-guided black-box model learning
GPs
Neural networks, etc

Fig. 8.9 Some modelling approaches on the white-black spectrum

rather than a single prediction point, allowing confidence intervals to be calculated and uncertainty to be propagated forward into any following analysis (see, e.g. [54]). As the use of GPs is now quite common, their complete formulation will not be introduced here. Suffice it to say that a GP requires a mean *function* $m(\underline{x})$ to be specified, rather than a constant vector, and a covariance function $k(\underline{x}, \underline{x}')$, rather than a matrix. The mean function expresses the central tendency of a process as it varies with time or space; the covariance expresses the variability. In the context of physics-informed machine learning, physical laws and constraints can be designed into the mean and covariance functions; in particular, for black-box models, the prior mean is often simply set to zero. The next case study illustrates how one can introduce physics into the mean function of a GP.

8.7.1 *Case Study: Prior Mean Functions—Residual Modelling*

The application domain here is in the performance monitoring of a cable-stayed bridge. The Tamar bridge is a cable-supported suspension bridge connecting Saltash and Plymouth in the South West of England, which has been monitored by the Vibration Engineering Section at the University of Exeter [55]. The interest here is in the development of a model to predict bridge-deck deflections that can be used as a performance indicator (see [56, 57]). The variations in deck deflections are driven by a number of factors, including fluctuating temperature and loading from traffic (which are included as inputs to the model). Figure 8.10a shows the regression target considered in this example, which is a longitudinal deflection. The monitoring period shown is from September (Autumn) to January (Winter). In this figure, one can see short-term fluctuations (daily), and a longer-term trend which is seasonal and driven by the increased hogging of the bridge deck as the ambient temperature decreases into the winter months. To mimic the situation where only a limited period of monitoring data is available for the establishment of an SHM algorithm, data from the initial month of the monitoring period are used to establish a GP regression model for deflection prediction (see [58] for more details).

A GP prediction, using the standard approach of a zero-mean prior, is shown in Fig. 8.10b. Here one can see exactly the behaviour that is expected; the model is able to predict the deck deflections well, in and around the training period, but is unable to predict the deflections in colder periods towards the end of the time series. The confidence intervals widen to reflect the fact that the inputs to the model towards the end of the period are different from those in the training set—this demonstrates the usefulness of the GP approach, as one knows not to trust the predictions in this period.

To formulate a grey-box model for this scenario, a physics-informed prior mean function is adopted that encodes the expected linear expansion behaviour of stay-cables with temperature [59]. Figure 8.10c shows the GP prediction with a linear prior mean function, where one can see a significant enhancement of predictive capability across the monitoring period. Where temperatures are at their lowest, the model predictions fall back on the prior mean function—which then

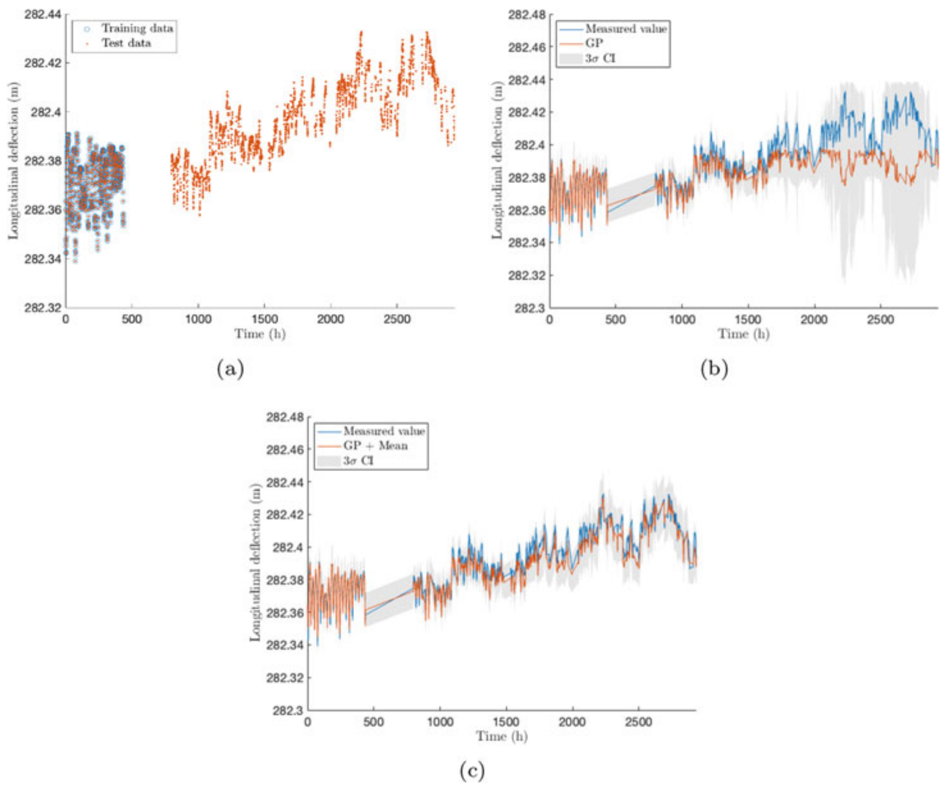

Fig. 8.10 Model for bridge deck deflections: (**a**) shows the training and test datasets for the GPs, (**b**) is a GP prediction with a zero mean function prior, and (**c**) shows the prediction when a simple physics-informed mean function is incorporated. See [58] for more details

behaves as a white box, allowing some extrapolative capability. The prediction error is significantly smaller for the grey-box model.

This example shows one of the significant strengths of a physics-informed approach. A nonparametric black-box learner will always be subject to generalisation issues; it will lose predictive capability if the conditions change from those experienced in the training set; it fails to extrapolate. In contrast, the grey-box model shows enhanced capability because it retains some explanatory power under any conditions in which the underlying physics hold.

Another advantage of grey-box models relates to the required quantity of the training data. If the white-box explains a significant part of the system behaviour, typically with a small number of meaningful parameters, this will reduce the necessary complexity of the black-box part. A simpler black-box model with a reduced parameter count then allows a reduction in the training data; this can be a significant advantage in engineering problems, where measured data may be expensive or difficult to obtain.

As mentioned above, using a GP as the nonparametric learner also allows physical insight to be included via the covariance function [60]. Although detailed case studies will not be presented here, the interested reader can consult [61] for a study relating to guided-wave structural health monitoring. In [62, 63], the physical boundary conditions for a plate structure are used to constrain GP predications on locating structural damage using acoustic emission signals.

8.8 Conclusions

This paper presents a tutorial treatment of certain data-based methods in nonlinear dynamics. The sheer size of the general subject has meant that only a small fraction of possible areas have been covered. The application domain within nonlinear dynamics has largely been restricted to the subject of *nonlinear system identification* (NLSI). Within NLSI, the choice was made to focus on Bayesian methodologies centred on white-box and grey-box models, although a large number of references have been cited, so that the interested reader can explore a little further afield. The paper also expands on the use of combining physics-based and data-based models via the technology provided in the burgeoning field of *physics-informed machine learning*.

Acknowledgments The authors gratefully acknowledge the help of their colleagues and friends in the research presented here: Anis Bin Abdessalem, Will Becker, James Brownjohn, Nikos Devilis, Ramon Fuentes, Sam Gibson, Marcus Haywood-Alexander, James Hensman, Geoff Holmes, Matty Jones, Ki-Young Koo, Graeme Manson, Rajdip Nayek, Dan Pitchforth, Tim Rogers and Sikai Zhang. KW would like to thank the Engineering and Physical Sciences Research Council (EPSRC) for funding from the Programme Grant EP/R006768/1. EJC would like to thank EPSRC for funding via the Innovation Fellowship EP/S001565/1. For the purpose of open access, the authors have applied a Creative Commons Attribution (CC BY) licence to any author accepted manuscript version arising.

References

1. Brunton, S.L., Kutz, J.N.: Data-Driven Science and Engineering: Machine Learning, Dynamical Systems and Control. Cambridge University Press, Cambridge (2019)
2. Bishop, C.M.: *Pattern Recognition and Machine Learning*. Springer, New York (2013)
3. Legendre, A.M.: Nouvelles méthodes pour la détermination des orbites des comètes. Didot, F., Paris (1805)
4. Stigler, S.M.: Gauss and the invention of least squares. Ann. Stat. **9**, 475–474 (1981)
5. Ljung, L.: System Identification: Theory for the User, 2nd edn. Prentice-Hall, Englewood Cliffs (1998)
6. Sodersröm, T., Stoica, P.: System Identification. Prentice-Hall, Englewood Cliffs (1994)
7. Worden, K., Tomlinson, G.R.: Nonlinearity in Structural Dynamics. Institute of Physics Press, New York (2001)
8. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley, Reading (1989)
9. Worden, K., Manson, G.: On the identification of hysteretic systems, Part I: fitness landscapes and evolutionary identification. Mech. Syst. Signal Process. **29**, 201–212 (2012)
10. Worden, K., Becker, W.E.: On the identification of hysteretic systems, Part II: Bayesian sensitivity analysis and parameter confidence. Mech. Syst. Signal Process. **29**, 213–227 (2012)
11. Beck, J.L., Katafygiotis, L.S.: Updating models and their uncertainties. I: Bayesian statistical framework. ASCE J. Eng. Mech. **124**, 455–461 (1998)
12. Beck, J.L., Au, S.-K.: Bayesian updating of structural models and reliability using Markov chain Monte Carlo simulation. ASCE J. Eng. Mech. **128**, 380–391 (2002)
13. Girolami, M.: Bayesian inference for differential equations. Theor. Comput. Sci. **408**, 4–16 (2008)
14. Worden, K., Hensman, J.J.: Parameter estimation and model selection for a class of hysteretic systems using Bayesian inference. Mech. Syst. Signal Process. **32**, 153–169 (2012)
15. Abdessalem, A.B., Dervilis, N., Wagg, D.J., Worden, K.: Model selection and parameter estimation in structural dynamics using approximate Bayesian computation. Mech. Syst. Signal Process. **99**, 306–325 (2018)
16. Kovacic, I., Brennan, M.J.: The Duffing Equation: Nonlinear Oscillators and their Behaviour. Wiley, New York (2011)
17. Wolf, A., Swift, J.B., Swinney, H.L., Vastano, J.A.: Determining Lyapunov exponents from a time series. Physica D: Nonlinear Phenomena **16**, 285–317 (1985)
18. Staszewski, W.J., Worden, K.: Wavelet analysis of time-series: coherent structures, chaos and noise. Int. J. Bifurcation Chaos **9**, 455–471 (1997)
19. Holmes, P., Lumley, J.L., Berkooz, G.: Turbulence, Coherent Structures, Dynamical Systems and Symmetry. Cambridge University Press, Cambridge (1996)
20. Calderhead, B., Girolami, M., Higham, D.J.: Is it safe to go out yet? Statistical inference in a zombie outbreak model. Technical report, University of Strathclyde, Department of Mathematics and Statistics, Glasgow (2010)
21. Gelman, A., Carlin, J.B., Stern, H.S., Rubin, D.B.: Bayesian Data Analysis, 2nd edn.. Chapman and Hall/CRC, London (2004)
22. Mackay, M.J.C.: Information Theory, Inference and Learning Algorithms. Cambridge University Press, Cambridge (2003)
23. The Mathworks. Matlab V7 (2004)
24. Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical Recipes: The Art of Scientific Computing, 3rd edn. Cambridge University Press, Cambridge (2007)
25. Patil, A., Huard, D., Fonnesbeck, C.J.: PyMC: Bayesian stochastic modelling in Python. J. Stat. Softw. **35**, 81 (2010)
26. Green, P.J.: Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. Biometrika **82**, 711–732 (1995)
27. Tiboaca, O.D., Green, P.L., Barthorpe, R.J., Worden, K.: Bayesian parameter estimation and model selection of a nonlinear dynamical system using reversible jump Markov chain Monte Carlo. In: Proceedings of 26th International Conference on Noise and Vibration Engineering, Leuven (2014)
28. Tiboaca, O.D., Green, P.L., Barthorpe, R.J., Antoniadou, I., Worden, K.: Bayesian inference and RJMCMC in structural dynamics—on experimental data. In: Proceedings of the 34rd International Modal Analysis Conference, Orlando, FL (2016)
29. Beaumont, M., Cornuet, J., Marin, J., Robert, C.: Adaptive approximate Bayesian computation. Biometrika **96**, 983–990 (2009)
30. Toni, T., Stumpf, M.P.H.: Simulation-based model selection for dynamical systems in systems and population biology. Bioinformatics **26**, 104–110 (2010)
31. Barnes, C., Silk, D., Stumpf, P.: Bayesian design strategies for synthetic biology. Interface Focus **1**, 895–908 (2011)
32. Turner, B.M., Van Zandt, T.: A tutorial on approximate Bayesian computation. J. Math. Psychol. **56**, 69–85 (2012)
33. Chiachio, M., Beck, J.L., Chiachio, J., Rus, G.: Approximate Bayesian computation by subset simulation. SIAM J. Sci. Comput. **36**, A1339–A1338 (2014)

34. Ching, J., Beck, J.L., Porter, K.: Bayesian state and parameter estimation of uncertain dynamical systems. Probab. Eng. Mech. **21**, 81–96 (2006)
35. Doucet, A., Godsill, S., Andrieu, C.: On sequential Monte Carlo sampling methods for Bayesian filtering. Stat. Comput. **10**, 197–208 (2000)
36. Filippi, S., Barnes, C.P., Cornebise, J., Stumpf, M.P.H.: On optimality of kernels for approximate Bayesian computation using sequential Monte Carlo. Stat. Appl. Genet. Mol. Biol. **12**, 87–107 (2013)
37. Bouc, R.: Forced vibration of mechanical system with hysteresis. In: Proceedings of 4th Conference on Nonlinear Oscillation, Prague (1967)
38. Wen, Y.K.: Method for random vibration of hysteretic systems. Proceedings of the American Society of Civil Engineers Journal of the Engineering Mechanics Division, pp. 102 (1976)
39. Wipf, D.P., Rao, B.D.: Sparse Bayesian learning for basis selection. IEEE Trans. Signal Process. **52**, 2153–2164 (2004)
40. Brunton, S.L., Proctor, J.L., Kutz, J.N.: Discovering governing equations from data by sparse identification of nonlinear dynamical systems. Proc. Natl. Acad. Sci. **113**, 3932–3937 (2016)
41. Fuentes, R., Dervilis, N., Worden, K., Cross, E.J.: Efficient parameter identification and model selection in nonlinear dynamical systems via sparse Bayesian learning. In: Journal of Physics: Conference Series, vol. 1264, pp. 012050 (2019)
42. Tipping, M.E.: Sparse Bayesian learning and the relevance vector machine. J. Mach. Learn. Res. **1**, 211–244 (2001)
43. Ishwaran, H., Rao, J.S.: Spike and slab variable selection: Frequentist and Bayesian strategies. Ann. Stat. **33**, 730–773 (2005)
44. Nayek, R., Fuentes, R., Worden, K., Cross, E.J.: On spike-and-slab priors for Bayesian equation discovery of nonlinear dynamical systems via sparse linear regression. Mech. Syst. Signal Process. **161**, 107986 (2021)
45. Kennedy, M., O'Hagan, A.: Bayesian calibration of computer models. J. R. Stat. Soc. Ser. B **63**, 425–464 (2005)
46. Pitchforth, D.J., Rogers, T.J., Tygesen, U.T., Cross, E.J.: Grey-box models for wave loading prediction. Mech. Syst. Signal Process. **159**, 107741 (2021)
47. Worden, K., Becker, W.E., Rogers, T.J., Cross, E.J.: On the confidence bounds of Gaussian process NARX models and their higher-order frequency response functions. Mech. Syst. Signal Process. **104**, 188–223 (2018)
48. Sohlberg, B.: Supervision and Control for Industrial Processes: Using Grey Box Models, Predictive Control and Fault Detection Methods. Springer Science and Business Media, Berlin (2012)
49. Sohlberg, B., Jacobsen, E.W.: Grey box modelling—branches and experiences. IFAC Proceedings Volumes **41**, 11415–11420 (2008)
50. Rogers, T.J., Holmes, G.R., Cross, E.J., Worden, K.: On a grey box modelling framework for nonlinear system identification. In: Special Topics in Structural Dynamics, vol. 6, pp. 167–178. Springer, Berlin (2017)
51. Worden, K., Barthorpe, R.J., Cross, E.J., Dervilis, N., Holmes, G.R., Manson, G., Rogers, T.J.: On evolutionary system identification with applications to nonlinear benchmarks. Mech. Syst. Signal Process. **112**, 194–232 (2018)
52. Willard, J., Jia, X., Xu, S., Steinbach, M., Kumar, V.: Integrating physics-based modeling with machine learning: A survey. arXiv preprint arXiv:2003.04919 (2020)
53. Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Learning. The MIT Press, New York (2006)
54. Gibson, S.J., Rogers, T.J., Cross, E.J.: Data-driven strain prediction models and fatigue damage accumulation. In: Proceedings of the 29th International Conference on Noise and Vibration Engineering (ISMA 2020) (2020)
55. Koo, K.Y., Brownjohn, J.M.W., List, D.I., Cole, R.: Structural health monitoring of the Tamar suspension bridge. Struct. Control. Health Monit. **20**, 609–625 (2013)
56. Cross, E.J., Worden, K., Koo, K.Y., Brownjohn, J.M.W.: Filtering environmental load effects to enhance novelty detection on cable-supported bridge performance. In: Bridge Maintenance, Safety and Management. CRC Press, New York (2012), pp. 745–752
57. Cross, E.J.: On Structural Health Monitoring in Changing Environmental and Operational Conditions. PhD thesis, University of Sheffield, Sheffield (2012)
58. Zhang, S., Rogers, T.J., Cross, E.J.: Gaussian process based grey-box modelling for SHM of structures under fluctuating environmental conditions. In: Proceedings of 10th European Workshop on Structural Health Monitoring (EWSHM 2020) (2020)
59. Westgate, R.: Environmental Effects on a Suspension Bridge's Performance. PhD thesis, University of Sheffield, Sheffield (2012)
60. Cross, E.J., Rogers, T.J.: Physics-derived covariance functions for machine learning in structural dynamics. In: SYSID 2021, Padova, Italy (2021)
61. Haywood-Alexander, M., Dervilis, N., Worden, K., Cross, E.J., Mills, R.S., Rogers, T.J.: Structured machine learning tools for modelling characteristics of guided waves. arXiv preprint arXiv:2101.01506 (2021)
62. Jones, M.R., Rogers, T.J., Gardner, P.A., Cross, E.J.: Constraining Gaussian processes for grey-box acoustic emission source localisation. In: Proceedings of the 29th International Conference on Noise and Vibration Engineering (ISMA 2020) (2020)
63. Jones, M.R., Rogers, T.J., Martinez, I.E., Cross, E.J.: Bayesian localisation of acoustic emission sources for wind turbine bearings. In: Health Monitoring of Structural and Biological Systems XV, International Society for Optics and Photonics, vol. 11593 (2021)