Probabilistic Forecasting and Bayesian Data Assimilation

Sebastian Reich and Colin Cotter

Contents

Preface

Classical mechanics is built upon the concept of *determinism*. Determinism means that knowledge of the current state of a mechanical system completely determines its future (as well as its past). During the 19th century, determinism became a guiding principle for advancing our understanding of natural phenomena, from empirical evidence to first principles and natural laws. In order to formalise the concept of determinism, the French mathematician Pierre Simon Laplace postulated an *intellect* now referred to as *Laplace's demon*:

> We may regard the present state of the universe as the effect of its past and the cause of its future. An intellect which at a certain moment would know all forces that set nature in motion, and all positions of all its items of which nature is composed, if this intellect were also vast enough to submit these data to analysis, it would embrace in a single formula the movements of the greatest bodies of the universe and those of the tiniest atoms; for such an intellect nothing would be uncertain and the future just like the past would be present before its eyes.[1]

Laplace's demon has three properties: (i) exact knowledge of the laws of nature; (ii) complete knowledge of the state of the universe at a particular point in time (of course, Laplace was writing in the days before knowledge of quantum mechanics and relativity); and (iii) the ability to solve any form of mathematical equation exactly. Except for extremely rare cases, none of these three conditions are met in practice. First, mathematical models generally provide a much simplified representation of nature. In the words of the statistician George Box: "All models are wrong, some are useful". Second, reality can only be assessed through measurements which are prone to measurement errors and which can only provide a very limited representation of the current state of nature. Third, most mathematical models cannot be solved analytically; we need to approximate them and then implement their solution on a computer, leading to further errors. At the end of the day, we might end up with a perfectly deterministic piece of computer code with relatively little correspondence to the evolution of the natural phenomena of interest to us.

[1] We have found this quote in the *Very Short Introduction to Chaos* by Smith (2007*b*), which has also stimulated a number of philosophical discussions on imperfect model forecasts, chaos, and data assimilation throughout this book. The original publication is *Essai philosophique dur les probabilités* (1814) by Pierre Simon Laplace.

Despite all these limitations, computational models have proved extremely useful, in producing ever more skillful *weather predictions*, for example. This has been made possible by an iterated process combining *forecasting* using highly sophisticated computational models, with *analysis* of model outputs using observational data. In other words, we can think of a computational weather prediction code as an extremely complicated and sophisticated device for extrapolating our (limited) knowledge of the present state of the atmosphere into the future. This extrapolation procedure is guided by a constant comparison of computer generated forecasts with actual weather conditions as they arrive, leading to subsequent adjustments of the model state in the weather forecasting system. Since both the extrapolation process and the data-driven model adjustments are prone to errors which can often be treated as *random*, one is forced to address the implied inherent *forecast uncertainties*. The two main computational tools developed within the meteorology community in order to deal with these uncertainties are *ensemble prediction* and *data assimilation*.

In ensemble prediction, forecast uncertainties are treated mathematically as random variables; instead of just producing a single forecast, ensemble prediction produces large sets of forecasts which are viewed as realisations of these random variables. This has become a major tool for quantifying uncertainty in forecasts, and is a major theme in this book. Meanwhile, the term "*data assimilation*" was coined in the computational geoscience community to describe methodologies for improving forecasting skill by combining measured data with computer generated forecasts. More specifically, data assimilation algorithms meld computational models with sets of observations in order to, *e.g.*, reduce uncertainties in the model forecasts or to adjust model parameters. Since all models are approximate and all data sets are partial snapshots of nature and are limited by measurement errors, the purpose of data assimilation is to provide estimates that are better than those obtained by using either computational models or observational data alone. While meteorology has served as a stimulus for many current data assimilation algorithms, the subject of uncertainty quantification and data assimilation has found widespread applications ranging from cognitive science to engineering.

This book focusses on the *Bayesian* approach to data assimilation and gives an overview over the subject by fleshing out key ideas and concepts, as well as explaining how to implement specific data assimilation algorithms. Instead of focussing on particular application areas, we adopt a general dynamical systems approach. More to the point, the book brings together two major strands of data assimilation: on the one hand, algorithms based on *Kalman's formulas* for Gaussian distributions together with their extension to nonlinear systems; and *sequential Monte Carlo methods* (also called *particle filters*) on the other. The common feature of all of these algorithms is that they use *ensemble prediction* to represent forecast uncertainties. Our discussion of ensemble-based data assimilation algorithms relies heavily on the *McKean approach* to filtering and the concept of *coupling of measures*, a well-established subject in probability which

has not yet found widespread applications to Bayesian inference and data assimilation. Furthermore, while data assimilation can formally be treated as a special instance of the mathematical subject of filtering and smoothing, applications from the geosciences have highlighted that data assimilation algorithms are needed for very high-dimensional and highly nonlinear scientific models where the classical large sample size limits of statistics cannot be obtained in practice. Finally, in contrast with the assumptions of the *perfect model scenario* (which are central to most of mathematical filtering theory), applications from geoscience and other application areas require data assimilation algorithms which can cope with systematic model errors. Hence robustness of data assimilation algorithms under finite ensemble/sample sizes and systematic model errors becomes of crucial importance. These aspects will also be discussed in this book.

It should have become clear by now that understanding data assimilation algorithms and quantification of uncertainty requires a broad array of mathematical tools. Therefore, the material in this book has to build upon a multidisciplinary approach synthesising topics from analysis, statistics, probability, and scientific computing. To cope with this demand we have divided the book into two parts. While most of the necessary mathematical background material on uncertainty quantification and probabilistic forecasting is summarised in Part I, Part II is entirely devoted to data assimilation algorithms. As well as classical data assimilation algorithms such as the Kalman filter, variational techniques, and sequential Monte Carlo methods, the book also covers newer developments such as the ensemble Kalman filter and ensemble transform filters. The *McKean approach* to sequential filtering in combination with coupling of measures serves as a unifying mathematical framework throughout Part II.

The book is written at an introductory level suitable for graduate students in applied mathematics, computer science, engineering, geoscience and other emerging application areas of Bayesian data assimilation. Although some familiarity with random variables and dynamical systems is helpful, necessary mathematical concepts are introduced when they are required. A large number of numerical experiments are provided to help to illustrate theoretical findings; these are mostly presented in a semi-rigorous manner. Matlab code for many of these is available via the book's wepage. Since we focus on ideas and concepts, we avoid proofs of technical mathematical aspects such as existence, convergence etc.; in particular, this is achieved by concentrating on finite-dimensional discrete time processes where results can be sketched out using finite difference techniques, avoiding discussion of Itô integrals, for example. Some more technical aspects are collected in appendices at the end of each chapter, together with descriptions of alternative algorithms that are useful but not key to the main story. At the end of each chapter we also provide exercises, together with a brief guide to related literature.

With probabilistic forecasting and data assimilation representing such rich and diverse fields, it is unavoidable that the authors had to make choices about the material to include in the book. In particular, it was necessary to omit many in-

teresting recent developments in *uncertainty quantification* which are covered by Smith (2014). A very approachable introduction to data assimilation is provided by Tarantola (2005). In order to gain a broader mathematical perspective, the reader is referred to the monograph by Jazwinski (1970), which still provides an excellent introduction to the mathematical foundation of filtering and smoothing. A recent, in-depth mathematical account of filtering is given by Bain & Crisan (2009). The monograph by del Moral (2004) provides a very general mathematical framework for the filtering problem within the setting of Feynman–Kac formulae and their McKean models. Theoretical and practical aspects of sequential Monte Carlo methods and particle filters can, for example, be found in Doucet, de Freitas & Gordon (2001). The popular family of *ensemble Kalman filters* is covered by Evensen (2006). The monograph by Majda & Harlim (2012) develops further extensions of the classic Kalman filter to imperfect models in the context of turbulent flows. We also mention the excellent monograph on optimal transportation and coupling of measures by Villani (2003).

We would like to thank: our colleagues Uri Ascher, Gilles Blanchard, Jochen Bröcker, Dan Crisan, Georg Gottwald, Greg Pavliotis, Andrew Stuart, and Peter Jan van Leeuwen for the many stimulating discussions centered around various subjects covered in this book; our students Yuan Cheng, Nawinda Chutsagulprom, Maurilio Gutzeit, Tobias Machewitz, Matthias Theves, James Tull, Richard Willis and Alexandra Wolff for their careful reading of earlier drafts of this book; Jason Frank, who provided us with detailed and very valuable feedback; Dan and Kate Daniels who provided childcare whilst much of Colin's work on the book was taking place; and David Tranah from Cambridge University Press who provided guidance throughout the whole process of writing this book.

Finally, we would like to thank our families: Winnie, Kasimir, Nepomuk, Rebecca, Matilda and Evan, for their patience and encouragement.

1 Prologue: how to produce forecasts

This chapter sets out a simplified mathematical framework that allows us to discuss the concept of forecasting and, more generally, prediction. Two key ingredients of prediction are: (i) we have a computational model which we use to simulate the future evolution of the physical process of interest given its current state;[1] and (ii) we have some measurement procedure providing partially observed data on the current and past states of the system. These two ingredients include three different types of error which we need to take into account when making predictions: (i) *precision errors* in our knowledge of the current state of the physical system; (ii) differences between the evolution of the computational model and the physical system, known as *model errors*; and (iii) *measurement errors* in the data that must occur since all measurement procedures are imperfect. Precision and model errors will both lead to a growing divergence between the predicted state and the system state over time, which we attempt to correct with data which has been polluted with measurement errors. This leads to the key question of data assimilation: *how can we best combine the data with the model to minimise the impact of these errors, and obtain predictions (and quantify errors in our predictions) of the past, present and future state of the system?*

1.1 Physical processes and observations

In this book we shall introduce *data assimilation algorithms*, and we shall want to discuss and evaluate their *accuracy* and *performance*. We shall illustrate this by choosing examples where the physical dynamical system can be represented mathematically. This places us in a somewhat artificial situation where we must generate data from some mathematical model and then pretend that we have only observed part of it. However, this will allow us to assess the performance of data assimilation algorithms by comparing our forecasts with the "true evolution" of the system. Once we have demonstrated the performance of such algorithms in this setting, we are ready to apply them to actual data assimilation problems

[1] It is often the case, in ocean modelling for example, that only partial observations are available and it is already challenging to predict the *current state* of the system (nowcasting). It is also often useful to reconstruct past events when more data becomes available (hindcasting).

where the true system state is unknown. This methodology is standard in the data assimilation community.

We shall use the term *surrogate physical process* to describe the model that we use to generate the true physical dynamical system trajectory for the purpose of these investigations. Since we are building the surrogate physical process purely to test out data assimilation algorithms, we are completely free to choose a model for this. To challenge these algorithms, the surrogate physical process should exhibit some complex dynamical phenomena. On the other hand, it should allow for numerically reproducible results so that we can make comparisons and compute errors. For example, we could consider a surrogate physical process described in terms of a finite-dimensional state variable $z \in \mathbb{R}^{N_z}$ of dimension $N_z \geq 1$, that has time dependence governed by an *ordinary differential equation* (ODE) of the form

$$\frac{\mathrm{d}z}{\mathrm{d}t} = f(z) + g(t), \quad z(0) = z_0, \tag{1.1}$$

with a chosen vector field $f : \mathbb{R}^{N_z} \to \mathbb{R}^{N_z}$ and a time-dependent function $g(t) \in \mathbb{R}^{N_z}$ for $t \geq 0$ such that solutions of (1.1) exist for all $t \geq 0$ and are unique. While such an ODE model can certainly lead to complex dynamic phenomena, such as *chaos*, the results are not easily reproducible since closed form analytic solutions rarely exist. Instead, we choose to replace (1.1) by a *numerical approximation* such as the *forward Euler scheme*

$$z^{n+1} = z^n + \delta t \left(f(z^n) + g(t_n) \right), \qquad t_n = n\,\delta t, \tag{1.2}$$

with iteration index $n \geq 0$, step-size $\delta t > 0$, and initial value $z^0 = z_0$.[2] Usually, (1.2) is used to approximate (1.1). However, here we will choose (1.2) to be our actual surrogate physical process with some specified value of δt (chosen sufficiently small for stability). This is then completely reproducible (assuming exact arithmetic, or a particular choice of rounding mode) since there is an explicit formula to obtain the sequence z^0, z^1, z^2, *etc.*

We shall often want to discuss time-continuous systems, and therefore we choose to use linear interpolation in between discrete time points t_n and t_{n+1},

$$z(t) = z^n + (t - t_n)\frac{z^{n+1} - z^n}{\delta t}, \quad t \in [t_n, t_{n+1}], \tag{1.3}$$

to obtain a completely reproducible time-continuous representation of a surrogate physical process. In other words, once the vector field f, together with the step-size δt, the initial condition z_0, and the forcing $\{g(t_n)\}_{n\geq 0}$, have been specified in (1.2), a unique function $z(t)$ can be obtained for $t \geq 0$, which we will denote by $z_{\mathrm{ref}}(t)$ for the rest of this chapter. It should be emphasised at this point that we need to pretend that $z_{\mathrm{ref}}(t)$ is not directly accessible to us during the data assimilation process. Our goal is to estimate $z_{\mathrm{ref}}(t)$ from partial

[2] Throughout this book we use superscript indices to denote a temporal iteration index, *e.g.* z^n in (1.2). Such an index should not be confused with the nth power of z. The interpretation of z^n should hopefully be clear from the circumstances of its use.

measurements of $z_{\rm ref}(t)$, using imperfect mathematical models of the dynamical system. We will return to these issues later in the chapter.

To clarify the setting, we next discuss a specific example for producing surrogate physical processes in the form of a reference solution $z_{\rm ref}(t)$.

Example 1.1 The *Lorenz-63 model* (Lorenz 1963) has a three-dimensional state variable $z := (\mathrm{x}, \mathrm{y}, \mathrm{z})^{\mathrm{T}} \in \mathbb{R}^{N_z}$, for scalar variables $\mathrm{x}, \mathrm{y}, \mathrm{z}$, with $N_z = 3$. The variable z satisfies an equation that can be written in the form (1.2) with vector field f given by

$$f(z) := \begin{pmatrix} \sigma(\mathrm{y} - \mathrm{x}) \\ \mathrm{x}(\rho - \mathrm{z}) - \mathrm{y} \\ \mathrm{xy} - \beta \mathrm{z} \end{pmatrix}, \tag{1.4}$$

and parameter values $\sigma = 10$, $\rho = 28$, and $\beta = 8/3$. We will use this vector field in the discrete system (1.2) to build a surrogate physical process with step-size $\delta t = 0.001$ and initial conditions

$$\mathrm{x}_0 = -0.587, \quad \mathrm{y}_0 = -0.563, \quad \mathrm{z}_0 = 16.870. \tag{1.5}$$

As we develop this example throughout this chapter, we will discuss model errors, defined as differences between the surrogate physical process and the imperfect model that we will use to make predictions. For that reason we include a non-autonomous forcing term g in (1.2), which will have different definitions in the two models. We shall define the forcing $g(t_n) = g^n = (g_1^n, g_2^n, g_3^n)^{\mathrm{T}} \in \mathbb{R}^3$ for the surrogate physical process as follows: set $a = 1/\sqrt{\delta t}$ and, for $n \geq 0$, define recursively

$$g_i^{n+1} = \begin{cases} 2g_i^n + a/2 & \text{if } g_i^n \in [-a/2, 0), \\ -2g_i^n + a/2 & \text{otherwise,} \end{cases} \tag{1.6}$$

for $i = 1, 2, 3$ with initial values

$$g_1^0 = a(2^{-1/2} - 1/2), \quad g_2^0 = a(3^{-1/2} - 1/2), \quad g_3^0 = a(5^{-1/2} - 1/2).$$

It should be noted that $g_i^n \in [-a/2, a/2]$ for all $n \geq 0$. In order to avoid an undesired accumulation of round-off errors in floating point arithmetic, we need to slightly modify the iteration defined by (1.6). A precise description of the necessary modification can be found in the appendix at the end of this chapter. A reader familiar with examples from the dynamical systems literature might have noticed that the iteration (1.6) reduces to the *tent map iteration* with $a = 1$ and the interval $[-1/2, 1/2]$ shifted to $[0, 1]$. The factor $a > 0$ controls the amplitude of the forcing and the interval has been shifted such that the forcing is centred about zero. We choose this for the surrogate physical process since it is completely reproducible in exact arithmetic, but has very complicated dynamics that can appear random.

The numerical solutions obtained from an application of (1.2) for $n = 0, \ldots, N-$

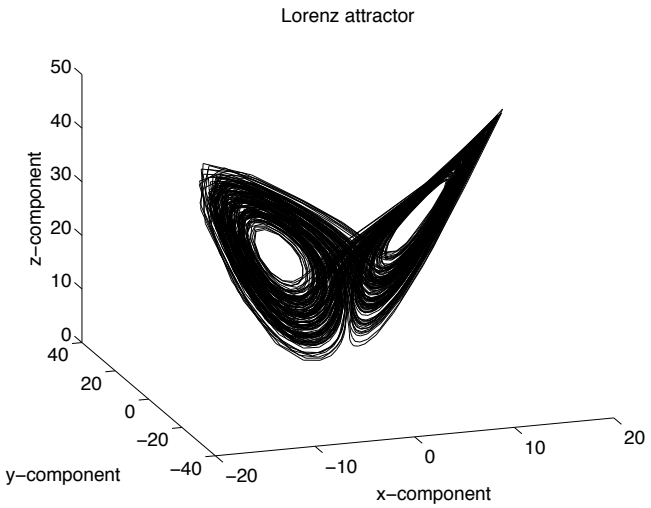

Figure 1.1 Trajectory of the modified Lorenz-63 model as described in Example 1.1. This trajectory provides us with the desired surrogate physical process. The cloud of solution points is part of what is called the model attractor.

1 with $N = 2 \times 10^5$ lead to a time continuous reference solution $z_{\mathrm{ref}}(t)$ according to the interpolation formula (1.3) for time $t \in [0, 200]$, which is used for all experiments conducted in this chapter. See Figure 1.1 for a phase portrait of the time series. Solutions asymptotically fill a subset of phase space $\mathbb{R}^3$ called the model *attractor*.

Next, we turn our attention to the second ingredient in the prediction problem, namely the measurement procedure. In this setting, neither $z_{\mathrm{ref}}(t)$ nor (1.2) will be explicitly available to us. Instead, we will receive "observations" or "measurements" of $z_{\mathrm{ref}}(t)$ at various times, in the form of measured data containing partial information about the underlying physical process, combined with measurement errors. Hence we need to introduce a mathematical framework for describing such partial observations of physical processes through measurements.

We first consider the case of an error-free measurement at a time t, which we describe by a *forward map* (or operator) $h : \mathbb{R}^{N_z} \to \mathbb{R}^{N_y}$

$$y_{\mathrm{obs}}(t) = h(z_{\mathrm{ref}}(t)), \tag{1.7}$$

where we typically have $N_y < N_z$ (corresponding to a partial observation of the system state z_{ref}). For simplicity, we shall only consider $N_y = 1$ in this chapter. Since h is non-invertible, we cannot deduce $z_{\mathrm{ref}}(t)$ from simple inversion, even if the measurements are free from errors.

More realistically, a measurement device will lead to measurement errors, which may arise as the linear superposition of many individual errors $\eta_i \in \mathbb{R}$,

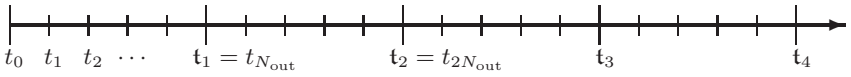

Figure 1.2 Diagram illustrating model timesteps t_0, t_1, *etc.* and observation times $\mathfrak{t}_1 = t_{N_{\rm out}}$, $\mathfrak{t}_2$, *etc.* Here, $N_{\rm out} = 5$.

$i = 1, \ldots, I$. Based on this assumption, we arrive at a mathematical model of type

$$y_{\rm obs}(t) = h(z_{\rm ref}(t)) + \sum_{i=1}^{I} \eta_i(t). \tag{1.8}$$

The quality of a measurement is now determined by the magnitude of the individual error terms η_i and the number I of contributing error sources. Measurements will only be taken at discrete points in time, separated by intervals of length $\Delta t_{\rm out} > 0$. To distinguish the discrete model time $t_n = n\,\delta t$ from instances at which measurements are taken, we use Gothic script to denote measurement points, *i.e.*,

$$\mathfrak{t}_k = k\,\Delta t_{\rm out}, \qquad k \geq 1,$$

and $\Delta t_{\rm out} = \delta t N_{\rm out}$ for given integer $N_{\rm out} \geq 1$. This is illustrated in Figure 1.2

We again consider a specific example to illustrate our "measurement procedure" (1.8).

Example 1.2 We consider the time series generated in Example 1.1 and assume that we can observe the x-component of

$$z_{\rm ref}(t) = (\mathrm{x}_{\rm ref}(t), \mathrm{y}_{\rm ref}(t), \mathrm{z}_{\rm ref}(t))^{\rm T} \in \mathbb{R}^3.$$

This leads to a linear forward operator of the form

$$h(z_{\rm ref}(t)) = \mathrm{x}_{\rm ref}(t).$$

In this example, we shall use a modified tent map of type (1.6) to model measurement errors. More specifically, we use the iteration

$$\xi_{k+1} = \begin{cases} 2\xi_k + a/2 & \text{if } \xi_k \in [-a/2, 0), \\ -2\xi_k + a/2 & \text{otherwise,} \end{cases} \tag{1.9}$$

with $a = 4$ and starting value $\xi_0 = a(2^{-1/2} - 1/2)$ for $k \geq 0$. From this sequence we store every tenth iterate in an array $\{\Xi_i\}_{i \geq 1}$, *i.e.*,

$$\Xi_i = \xi_{k=10i}, \qquad i = 1, 2, \ldots \tag{1.10}$$

An observation $\mathrm{x}_{\rm obs}$ at time $\mathfrak{t}_1 = \Delta t_{\rm out} = 0.05$ is now obtained as follows:

$$\mathrm{x}_{\rm obs}(\mathfrak{t}_1) := \mathrm{x}_{\rm ref}(\mathfrak{t}_1) + \frac{1}{20} \sum_{i=1}^{20} \Xi_i.$$

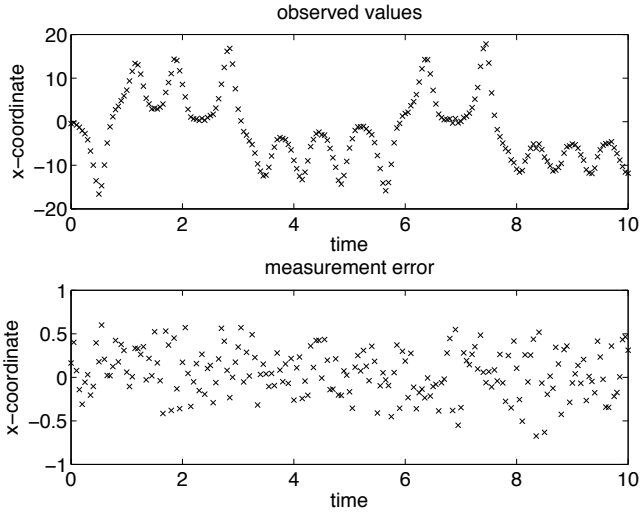

Figure 1.3 Observed values for the x-component and their measurement errors over the time interval $[0, 10]$ with observations taken every $\Delta t_{\text{out}} = 0.05$ time units.

This procedure fits into the framework of (1.8) with $I = 20$ and $\eta_i(\mathrm{t}_1) = \Xi_i/20$, $i = 1, \ldots, 20$.

For the next observation at $\mathrm{t}_2 = 2\Delta t_{\text{out}} = 0.1$ we use

$$\mathrm{x}_{\text{obs}}(\mathrm{t}_2) = \mathrm{x}_{\text{ref}}(\mathrm{t}_2) + \frac{1}{20}\sum_{i=21}^{40} \Xi_i,$$

and this process is repeated for all available data points from the reference trajectory generated in Example 1.1. Numerical results are displayed for the first 200 data points in Figure 1.3. Our procedure of defining the measurement errors might appear unduly complicated, but we will find later in Chapter 2 that it mimics important aspects of typical measurement errors. In particular, the measurement errors can be treated as *random* even though a perfectly deterministic procedure has defined them.

1.2 Data driven forecasting

We now assume that N_{obs} scalar observations $y_{\text{obs}}(\mathrm{t}_k) \in \mathbb{R}$ at $\mathrm{t}_k = k\,\Delta t_{\text{out}}$, $k = 1, 2, \ldots, N_{\text{obs}}$, have been made at time intervals of Δt_{out}. To define what we understand by a *forecast* or a *prediction*, we select a point in time t_{k_*} that we denote the *present*. Relative to t_{k_*}, we can define the *past* $t < \mathrm{t}_{k_*}$ and the *future* $t > \mathrm{t}_{k_*}$. A possible forecasting (or prediction) problem would be to produce an

estimate for

$$y_{\mathrm{ref}}(t) := h(z_{\mathrm{ref}}(t))$$

with $t > \mathfrak{t}_{k_*}$ and only observations from the past and present available. Such statements can be *verified* as soon as a future moment becomes the present and a new measurement becomes available. More generally, we would, of course, like to make predictions about the complete surrogate process $z_{\mathrm{ref}}(t)$ for $t > \mathfrak{t}_{k_*}$ and not only about the quantity we can actually observe. We will come back to this more challenging task later in this chapter.

Returning to the problem of predicting future observations, we first utilise the concept of *polynomial interpolation*. Recall that there is a unique polynomial

$$q(t) = b_0 + b_1 t + b_2 t^2 + \cdots + b_p t^p \tag{1.11}$$

of order p with coefficients b_l through any $p+1$ data points. We would like to find a polynomial that interpolates observations at $p+1$ present and past observation times $\{\mathfrak{t}_{k*}, \mathfrak{t}_{k*-1}, \cdots, \mathfrak{t}_{k*-p}\}$ with the aim of using it to predict future observations. This leads to the interpolation conditions

$$q(\mathfrak{t}_k) = y_{\mathrm{obs}}(\mathfrak{t}_k), \qquad \mathfrak{t}_k \in \{\mathfrak{t}_{k_*}, \mathfrak{t}_{k_*-1}, \cdots \mathfrak{t}_{k_*-p}\},$$

which determine the $p+1$ coefficients b_l in (1.11) uniquely. A predicted observation at $t > \mathfrak{t}_{k_*}$ is then simply provided by $q(t)$. Since t is outside the interval of the observed data points, the prediction is an *extrapolation* from the data. For the linear case $p = 1$ we obtain:

$$\begin{aligned} q(t) &= y_{\mathrm{obs}}(\mathfrak{t}_{k_*}) + (t - \mathfrak{t}_{k_*}) \frac{y_{\mathrm{obs}}(\mathfrak{t}_{k_*}) - y_{\mathrm{obs}}(\mathfrak{t}_{k_*-1})}{\mathfrak{t}_{k_*} - \mathfrak{t}_{k_*-1}} \\ &= y_{\mathrm{obs}}(\mathfrak{t}_{k_*}) + (t - \mathfrak{t}_{k_*}) \frac{y_{\mathrm{obs}}(\mathfrak{t}_{k_*}) - y_{\mathrm{obs}}(\mathfrak{t}_{k_*} - \Delta t_{\mathrm{out}})}{\Delta t_{\mathrm{out}}}. \end{aligned}$$

Upon setting $t = \mathfrak{t}_{k_*+1}$ we obtain the extrapolation formula

$$y_{\mathrm{predict}}(\mathfrak{t}_{k_*+1}) := q(\mathfrak{t}_{k_*+1}) = 2y_{\mathrm{obs}}(\mathfrak{t}_{k_*}) - y_{\mathrm{obs}}(\mathfrak{t}_{k_*-1}). \tag{1.12}$$

As soon as $y_{\mathrm{obs}}(\mathfrak{t}_{k_*+1})$ becomes available, we can compare this prediction with the observed value. Furthermore, we can use this new observation point (and discard the oldest one from $\mathfrak{t}_{k*-1}$) and a correspondingly updated linear extrapolation formula to obtain $y_{\mathrm{predict}}(\mathfrak{t}_{k_*+2})$. This can be iterated over several time intervals, repeatedly using data to predict the new observation. To assess the accuracy of this procedure we introduce the following measure.

Definition 1.3 (Root mean square error) For a set of predictions and observations at times $\{\mathfrak{t}_1, \mathfrak{t}_2, \cdots, \mathfrak{t}_N\}$ the root mean square error (RMSE) is given by

$$\text{time averaged RMSE} = \sqrt{\frac{1}{N} \sum_{k=1}^{N} |y_{\mathrm{obs}}(\mathfrak{t}_k) - y_{\mathrm{predict}}(\mathfrak{t}_k)|^2}. \tag{1.13}$$

In the case of linear interpolation, if there are N_{obs} observations then $N = N_{\text{obs}} - 2$ since we cannot make predictions using linear interpolation for the first two observations.

We illustrate the linear interpolation prediction strategy by our next example.

Example 1.4 We utilise the observations generated in Example 1.2 for the first solution component of the Lorenz-63 system, *i.e.*, $y_{\text{obs}}(\mathrm{t}_k) = \mathrm{x}_{\text{obs}}(\mathrm{t}_k)$. Recall that the observation interval is $\Delta t_{\text{out}} = 0.05$. We set the first t_{k_*} equal to $\mathrm{t}_{k_*} = 100$, and make a total of 2000 verifiable predictions until we reach $t = 200$. The linear extrapolation formula (1.12) is used for making predictions of observations, and the quality of these predictions is assessed using the time averaged RMSE (1.13) with $N = 2000$. A snapshot of the computational results over a short time-window can be found in Figure 1.4. The time averaged RMSE over the whole interval is approximately 1.2951.

It is usually desirable to "extend the prediction window" by making predictions further into the future. In view of this, we modify the procedure so that at each time t_{k_*}, we attempt to predict the observation at time t_{k_*+2} instead of t_{k*+1}. The associated linear extrapolation formula becomes

$$y_{\text{predict}}(\mathrm{t}_{k_*+2}) := q(\mathrm{t}_{k_*+2}) = 3y_{\text{obs}}(\mathrm{t}_{k_*}) - 2y_{\text{obs}}(\mathrm{t}_{k_*-1}).$$

The results can also be found in Figure 1.4; the quality of the predictions is clearly worse over this larger window. This is confirmed by the time averaged RMSE which increases to approximately 3.3654.

The results of Example 1.4 show that linear interpolation does not provide good predictions over longer times. This suggests to improve the accuracy of forecasts by extending the extrapolation formula (1.12) to use a linear combination of the present data point plus several previous data points of the form

$$y_{\text{predict}}(\mathrm{t}_{k_*+1}) = \sum_{l=0}^{p} a_l \, y_{\text{obs}}(\mathrm{t}_{k_*-l}). \tag{1.14}$$

We have already seen that linear extrapolation fits into this framework with $p = 1$ and coefficients $a_0 = 2$, $a_1 = -1$. We recall that the linear extrapolation formula (1.12) was based on first deriving the linear interpolation formula. Hence, as a first attempt at deriving coefficients a_l for (1.14) with $p > 1$, we shall use higher-order interpolation formulas. Interpolation formulas of order p can be conveniently based on the Lagrange polynomials (Süli & Mayers 2006) of order p

$$l_j(t) = \frac{\prod_{i \neq j}(t - \mathrm{t}_i)}{\prod_{i \neq j}(\mathrm{t}_j - \mathrm{t}_i)},$$

where the indices i and j run over the integers

$$\{k_*, k_* - 1, \ldots, k_* - p\}.$$

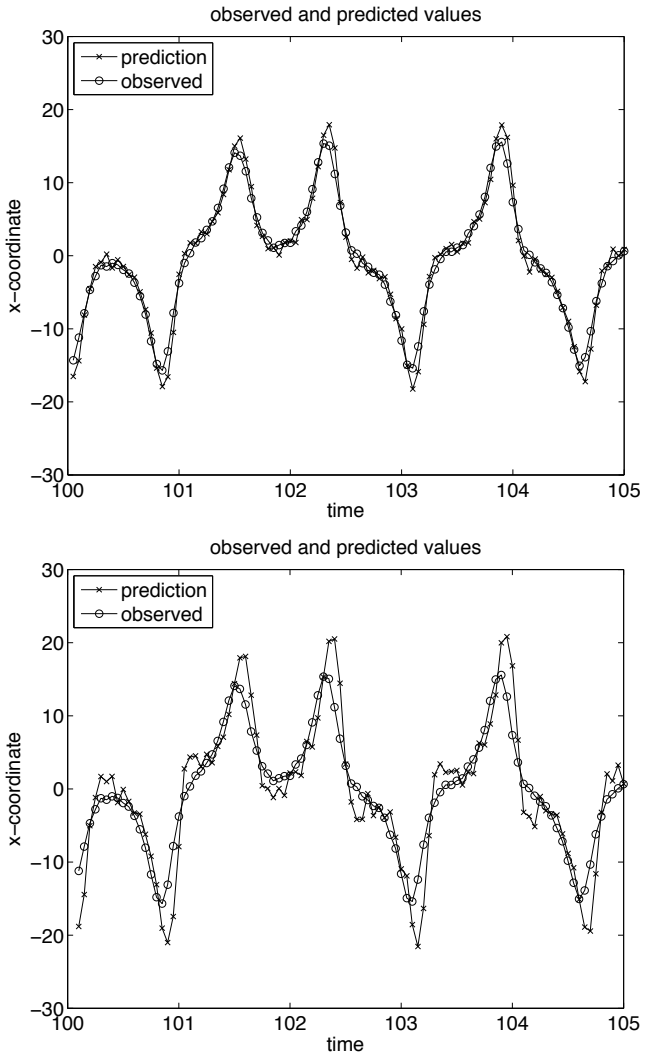

Figure 1.4 Observed values for the x-component and its predicted values using linear extrapolation. The figure at the top shows the results from linear extrapolation over a single observation interval $\Delta t_{\text{out}} = 0.05$, while the figure beneath shows results when doubling the prediction interval to 0.1 time units.

These polynomials have the useful property that

$$l_j(\mathrm{t}_i) = \begin{cases} 1 & \text{if } j = i, \\ 0 & \text{otherwise,} \end{cases}$$

which leads to the interpolation formula

$$q(t) = l_{k_*}(t)\, y_{\text{obs}}(\mathrm{t}_{k_*}) + l_{k_*-1}(t)\, y_{\text{obs}}(\mathrm{t}_{k_*-1}) + \cdots + l_{k_*-p}(t)\, y_{\text{obs}}(\mathrm{t}_{k_*-p}). \quad (1.15)$$

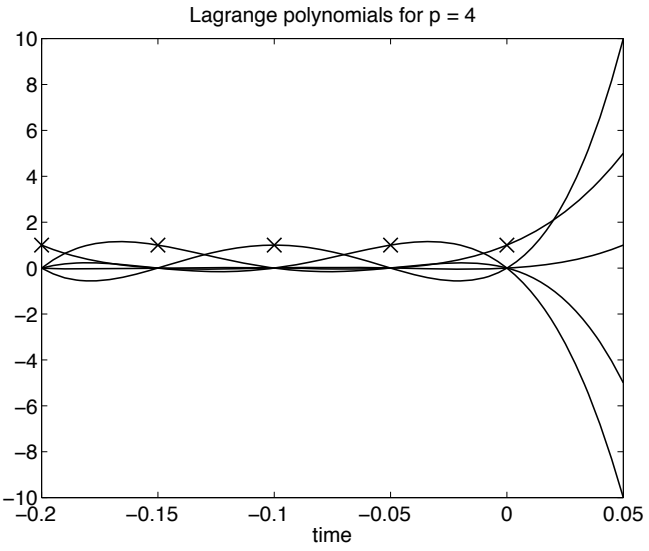

Figure 1.5 Lagrange polynomials $l_j(t)$ of order four corresponding to observations at $t_i = 0, -0.05, -0.1, -0.15, -.20$. The coefficients a_l in (1.14) are equal to the values of the Lagrangian polynomials at $t = 0.05$. Crosses mark the points where each polynomial takes the value one. Note that the other polynomials are zero at those interpolation points, and note the steep increase in magnitude outside the interpolation interval $t \in [-0.2, 0]$.

The coefficients a_l in (1.14) are obtained by setting $t = \mathrm{t}_{k_*+1}$ in (1.15), *i.e.*

$$a_l = l_{k_*-l}(\mathrm{t}_{k_*+1}), \qquad l = 0, 1, \ldots, p.$$

Example 1.5 We consider extrapolation based on polynomial interpolation of order $p = 4$. The associated extrapolation coefficients in (1.14) are

$$a_0 = 5,\ a_1 = -10,\ a_2 = 10,\ a_3 = -5,\ a_4 = 1,$$

and the associated Lagrange polynomials are shown in Figure 1.5, taking $\mathrm{t}_{k_*} = 0$ for simplicity. The values of the extrapolation coefficients can be obtained by inspecting the intersection of the Lagrange polynomials with the vertical line at $t = 0.05$.

The results of applying the fourth-order extrapolation formula to the data set from Example 1.2 are shown in Figure 1.6; the time averaged RMSE was 4.2707. This error is much larger than that observed for linear extrapolation (compare Example 1.4). The reason for this discrepancy can be found in the strong separation of the Lagrange polynomials outside the interpolation interval (compare Figure 1.5), which results in relatively large coefficients a_l in (1.14). Hence even relatively small measurement errors can get severely amplified and do not necessarily cancel out. This effect becomes even more pronounced when the

prediction interval is doubled to $2\Delta t_{\text{out}}$. The associated extrapolation coefficients are now given by

$$a_0 = 15,\ a_1 = -40,\ a_2 = 45,\ a_3 = -24,\ a_4 = 5.$$

See Figure 1.6 for numerical results.

We now discuss an entirely different approach for determining the coefficients a_l in (1.14). Instead of using polynomial interpolation, we shall seek the coefficients that optimise the prediction errors for a chosen subset of the observations, which we call the *training set*. These extrapolation coefficients are then fixed, and can be used to predict future observations. We shall assess the performance of our extrapolation coefficients on the remaining observations points, which we shall call the *test set*. For simplicity, let us assume that the training set consists of the first $N_{\text{T}} < N_{\text{obs}}$ observations $\{y_{\text{obs}}(\mathfrak{t}_1), \dots, y_{\text{obs}}(\mathfrak{t}_{N_{\text{T}}})\}$, and use the remaining data points as the test set.

Given a chosen set of coefficients $a_l \in \mathbb{R}$, $l = 0, \dots, p$, we can obtain a prediction of $y_{\text{obs}}(\mathfrak{t}_{j+p+1})$ for $0 < j \leq N_{\text{T}} - p - 1$ by using (1.14). The quality of the predictions is measured by the residuals

$$\begin{aligned} r_j &= y_{\text{obs}}(\mathfrak{t}_{j+p+1}) - y_{\text{predict}}(\mathfrak{t}_{j+p+1}) \\ &= y_{\text{obs}}(\mathfrak{t}_{j+p+1}) - \sum_{l=0}^{p} a_l y_{\text{obs}}(\mathfrak{t}_{j+p-l}) \end{aligned} \tag{1.16}$$

for $j = 1, 2, \dots, J$ with $J = N_{\text{T}} - p - 1$. We now seek the coefficients a_l in (1.14) such that the resulting time averaged RMSE is minimised over the training set. This is equivalent to minimising the functional

$$L(\{a_l\}) = \frac{1}{2} \sum_{j=1}^{J} r_j^2;$$

we have recovered the *method of least squares*. The minimum of $L(\{a_l\})$ is attained when the partial derivatives of L with respect to the coefficients a_l vanish, *i.e.*,

$$\frac{\partial L}{\partial a_l} = -\sum_{j=1}^{J} y_{\text{obs}}(\mathfrak{t}_{j+p-l})\, r_j = 0 \tag{1.17}$$

for $l = 0, \dots, p$. These conditions lead to $p + 1$ linear equations which may be solved for the $p + 1$ unknown coefficients a_l.

Once an optimal set of coefficients a_l has been found, these coefficients can be used in (1.14) to make predictions over the test set. The underlying assumption is that the training and test sets display a similar behaviour. Mathematically speaking, this relies on the assumption of *stationarity* of the time series of observations. See Chorin & Hald (2009) for more details.

We mention in passing that (1.14) with coefficients a_l determined by the

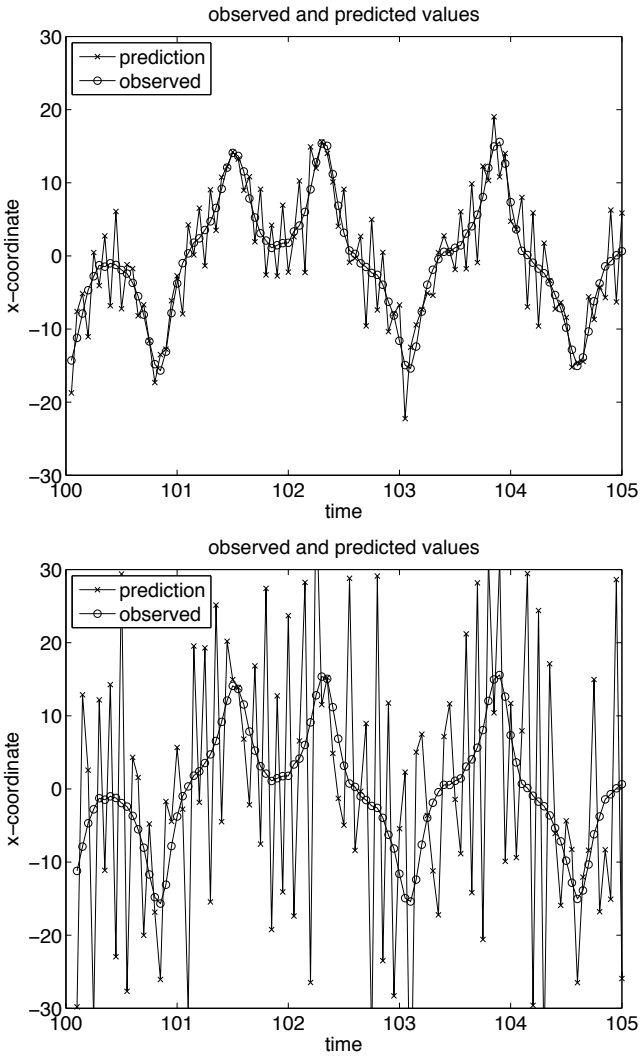

Figure 1.6 Observed values for the x-component and its predicted values using fourth-order extrapolation. The figure at the top shows the results from extrapolation over a single observation interval $\Delta t_{\text{out}} = 0.05$ while the figure beneath shows results for doubling the prediction interval to 0.1 time units. Compare these results to the ones displayed in Figure 1.4 for linear extrapolation.

method of least squares may be considered as a particular instance of an *autoregressive model* of order $p+1$. The class of autoregressive models provides an example of purely data driven models.

Example 1.6 We return again to the setting from Example 1.4. We replace the linear extrapolation procedure by predictions using (1.14) with the coefficients a_l

determined by the method of least squares. We find that setting $p = 4$ in (1.14) and a training set with $N_{\mathrm{T}} = N_{\mathrm{obs}}/2 = 2000$ leads to a time averaged RMSE of 0.9718 with coefficients

$$a_0 = 2.0503,\ a_1 = -1.2248,\ a_2 = -0.2165,\ a_3 = 0.4952,\ a_4 = -0.1397.$$

Note that these values fluctuate much less about the observed values than those obtained from fourth-order interpolation (compare Example 1.5) and that the values for a_0 and a_1 are relatively close to those obtained from linear extrapolation (compare (1.12)). Hence we may argue that the method of least squares leads to a modified linear extrapolation procedure with a slight reduction in the time averaged RMSE.

We also apply the same methodology to predict y at t_{k_*+2} (prediction over $2\Delta t_{\mathrm{out}} = 0.01$) and find that the averaged RMSE increases to 2.3039. See Figure 1.7 for some numerical results.

The mathematical structure of the least squares approach becomes more transparent when put into matrix notation. We first collect the unknown coefficients a_l into a vector

$$x = \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_p \end{pmatrix} \in \mathbb{R}^{p+1}$$

and the residuals r_j into a vector

$$r = \begin{pmatrix} r_1 \\ r_2 \\ \vdots \\ r_J \end{pmatrix} \in \mathbb{R}^J.$$

Next we write (1.14) as

$$r = b - Ax,$$

where $b \in \mathbb{R}^J$ is defined by

$$b = \begin{pmatrix} y_{\mathrm{obs}}(\mathrm{t}_{p+2}) \\ y_{\mathrm{obs}}(\mathrm{t}_{p+3}) \\ \vdots \\ y_{\mathrm{obs}}(\mathrm{t}_{p+J+1}) \end{pmatrix} \in \mathbb{R}^J$$

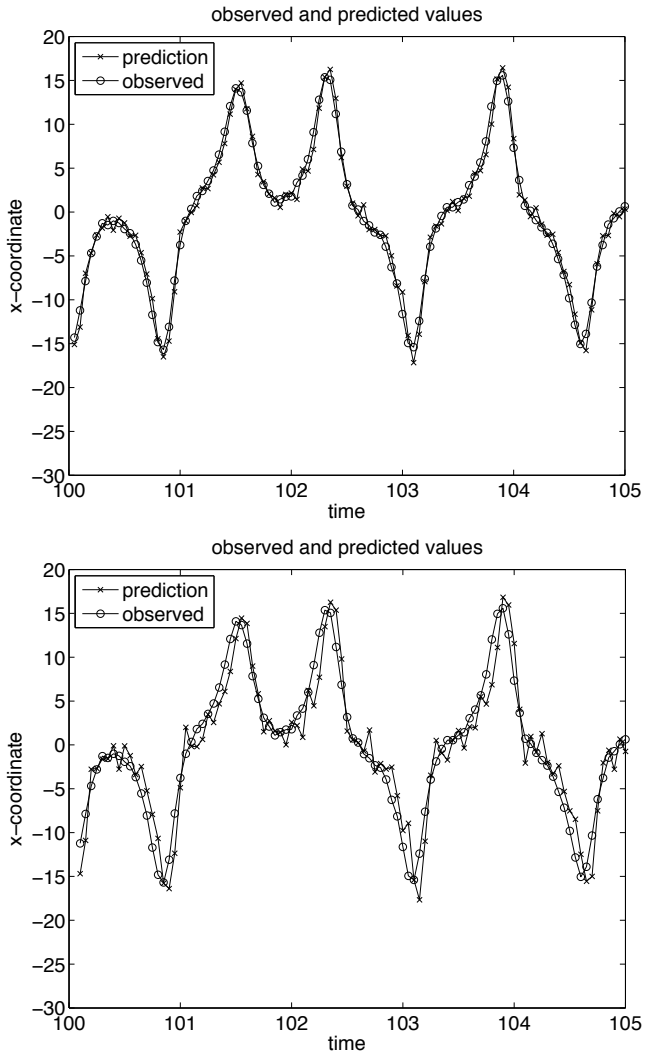

Figure 1.7 Observed values of the x-component and corresponding predicted values using the method of least squares with $p = 4$ in (1.14). The figure on the top shows the results from predictions over a single observation interval $\Delta t_{\text{out}} = 0.05$, and the figure beneath shows results when doubling the prediction interval to 0.1 time units. The results should be compared to those in Figures 1.4 and 1.6 obtained from extrapolation.

and the matrix $A \in \mathbb{R}^{J\times(p+1)}$ by

$$A = \begin{pmatrix} y_{\text{obs}}(\mathrm{t}_{p+1}) & y_{\text{obs}}(\mathrm{t}_p) & \cdots & y_{\text{obs}}(\mathrm{t}_1) \\ y_{\text{obs}}(\mathrm{t}_{p+2}) & y_{\text{obs}}(\mathrm{t}_{p+1}) & \cdots & y_{\text{obs}}(\mathrm{t}_2) \\ \vdots & \vdots & & \vdots \\ y_{\text{obs}}(\mathrm{t}_{p+J}) & y_{\text{obs}}(\mathrm{t}_{p+J-1}) & \cdots & y_{\text{obs}}(\mathrm{t}_J) \end{pmatrix}.$$

For square matrices A (*i.e.*, $J = p + 1$) with $\det(A) \neq 0$ we recall that x can be determined such that $r = 0$ and

$$x = A^{-1}b.$$

However, in practice we are usually dealing with the case of overdetermined systems of equations for which $J \gg p + 1$ and for which there is in general no x such that $r = 0$. The method of least squares determines $x_* \in \mathbb{R}^{p+1}$ such that the norm of the vector r is minimised, *i.e.*,

$$x_* = \arg\min \|r\|^2 = \arg\min \|Ax - b\|^2.$$

We find that the gradient of the functional $L(x) = \|Ax - b\|^2$ with respect to x is given by

$$\nabla_x L(x) = 2A^{\mathrm{T}}(Ax - b) \in \mathbb{R}^{p+1}.$$

Here $A^{\mathrm{T}} \in \mathbb{R}^{(p+1)\times J}$ denotes the transpose of A. Furthermore, the Hessian (matrix of second derivatives) of $L(x)$ is

$$H = 2A^{\mathrm{T}}A \in \mathbb{R}^{(p+1)\times(p+1)},$$

which is positive definite when A has maximum column rank $p + 1$. If this is the case, then setting $\nabla L(x_*) = 0$ leads to the following equation for x_*,

$$A^{\mathrm{T}}Ax_* = A^{\mathrm{T}}b.$$

We can confirm that we have a minimiser of $L(x)$ since

$$\begin{aligned} L(x_* + \delta x) &= L(x_*) + \nabla_x L(x_*)^{\mathrm{T}}\delta x + \delta x^{\mathrm{T}}A^{\mathrm{T}}A\delta x, \\ &= L(x_*) + \delta x^{\mathrm{T}}A^{\mathrm{T}}A\delta x, \\ &> L(x_*), \end{aligned}$$

for all vectors $\delta x \neq 0$.

1.3 Model driven forecasting and data assimilation

So far in this chapter, we have used observations and elementary mathematical tools to design *linear models* for predicting future outcomes in the observable variable $y = h(z)$. More precisely, we have considered mathematical tools that rely on the observed quantities alone, without any reference to our surrogate physical process from which they were generated. The predictions were constrained by the assumption of a polynomial form in time, or by optimising the coefficients over a training set. These models are often described as *empirical* or *bottom-up*. We now introduce our third ingredient, the use of *mechanistic* or *top-down* models of the physical process that are derived from *first principles*, a process well established in the context of classical mechanics (Arnold 1989), for example. In practice such first principles might be provided by conservation of mass and/or energy or by Newton's laws of motion, or other analogues in *e.g.*

biology, sociology or economics. Given an estimate of the system state $z(t_0)$ at time t_0, a model allows us to obtain estimates of the system state $z(t)$ for $t > t_0$. In almost all cases the model is imperfect, and model errors lead to increased errors in the state estimate over time, unless it is corrected by introducing more data at later times.

In the somewhat artificial setting of this chapter, we imagine that understanding of the surrogate physical process has allowed us to derive a model from first principles, in the form of the difference equation

$$z^{n+1} = z^n + \delta t f(z^n), \qquad t_{n+1} = t_n + \delta t. \tag{1.18}$$

In this case, we have chosen a scenario where the difference between the surrogate physical process, as provided by (1.2), and our mechanistic model, as given by (1.18), is simply in the inclusion or omission of the time-dependent driving term $g(t)$. This allows us to easily quantify the impact of the error. In practice, when data is obtained from an observed physical process, quantifying this error is a much more difficult problem. In our case, provided that we have exact knowledge of the state z^n_{ref} of our surrogate physical process at time t_n and provided we use this information in order to set $z^n_{\text{model}} = z^n_{\text{ref}}$ in our mechanistic model, the one step ahead *prediction error* $e^{n+1} = z^{n+1}_{\text{model}} - z^{n+1}_{\text{ref}}$ is given by

$$e^{n+1} = -\delta t\, g(t_n), \qquad t_n = n\,\delta t. \tag{1.19}$$

We will also call e^n the *model error* since e^n reflects the difference between (1.2), our surrogate physical process, and (1.18), our mechanistic model for this process. One specific type of model errors are *discretisation errors* that arise when mechanistic models are approximated numerically. We will return to the issue of discretisation errors in Chapter 4.

At this point two major challenges arise. First, we wish to predict over time intervals much larger than δt. Second, we can only partially observe the present states of the underlying physical process in intervals of Δt_{out}; we do not have access to the full state vector $z_{\text{ref}}(t)$ at any moment in time. The first difficulty requires us to assess the propagation and accumulation of model errors over several timesteps, under the hypothetical assumption that both the physical process and the mechanistic model start from the same initial state z^0 at $t_0 = 0$.

We explore this in the next example.

Example 1.7 We return to Example 1.1 and simulate (1.18) with the vector field $f(z)$ given by (1.4). We then compare the surrogate physical process as simulated in Example 1.1.

The numerical solution obtained from an application of (1.18) is stored over a time-interval $\mathfrak{t}_0 = 0$ to $\mathfrak{t}_{\text{end}} = 200$ in intervals of $\Delta t_{\text{out}} = 0.05$. These 3×4001 data points provide us with the model output $z_{\text{model}}(\mathfrak{t}_k)$ at $\mathfrak{t}_k = k\,\Delta t_{\text{out}}$, which can be compared to the reference trajectory $z_{\text{ref}}(\mathfrak{t}_k)$ from Example 1.1. We plot the phase portrait of the time series from our mechanistic model in Figure 1.8. The result looks rather similar to the phase portrait displayed in Figure 1.1, which

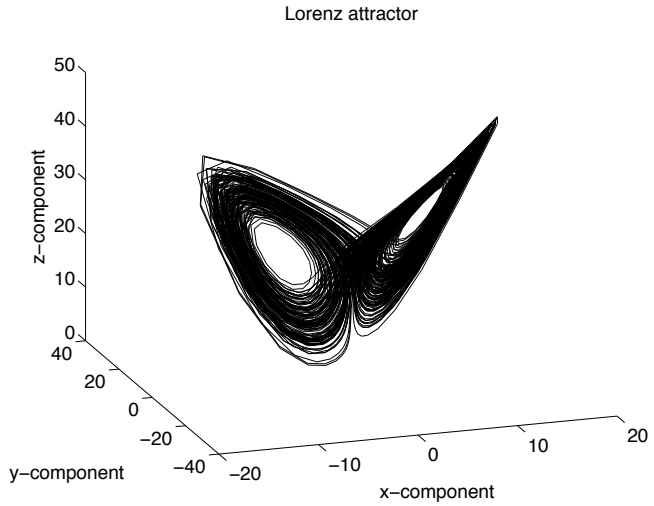

Figure 1.8 Long trajectory from our mechanistic model which traces the Lorenz attractor of our model. The shape of the attractor is nearly identical to what is displayed in Figure 1.1.

indicates that our mechanistic model is able to capture qualitative aspects of the surrogate physical process.

We next check whether this property carries over to specific predictions. In order to assess this aspect of our mechanistic model, both the mechanistic model and physical process are started from the same initial condition (1.5) at time $t_0 = 0$. We display the results in all three state variables over a time interval $[0, 10]$ in Figure 1.9. A difference in the solutions becomes noticeable at about $t = 2$; this is much longer than the prediction intervals obtained for linear interpolation and/or an autoregressive model obtained from the method of least squares. However, this comparison is unrealistic as we require the precise knowledge of the initial state in order to make predictions based on our mechanistic model. Indeed, the exact initial or present state is unavailable in most practical applications.

The previous example has demonstrated that the use of mechanistic models can lead to skillful predictions over relatively long time intervals, provided the model state from which we start our prediction is sufficiently close to the state of the physical process under consideration at the initial time. It should also be obvious that the quality of model-based predictions will depend on the relative magnitude of the modeling errors e^n and the subsequent systematic contributions from $\delta t f(z^n)$ in (1.18).

From these findings we conclude that (i) we need methods for estimating appropriate initial conditions for our mechanistic model from the available observations; and that (ii) we should strive to improve our mechanistic models by

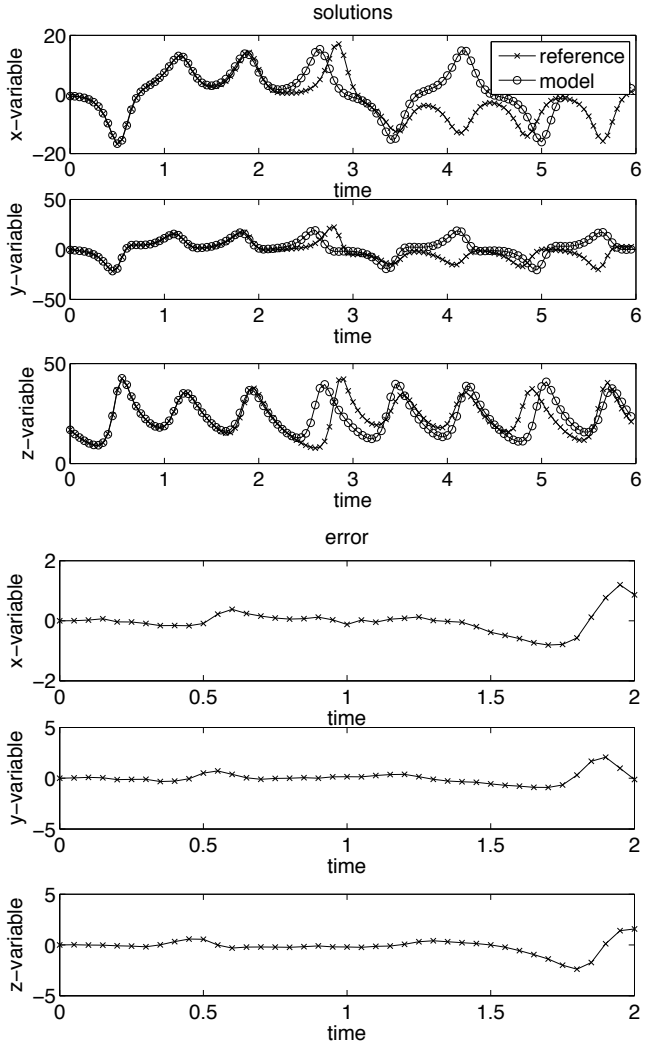

Figure 1.9 We compare the behaviour of our mechanistic model to the reference trajectory under the assumption that both start from the same initial condition at time $t = 0$. The differences between the model and nature are caused by the non-autonomous driving terms $g^n = g(t_n)$ in (1.2) and their accumulative effect. These differences become significant at about $t = 2$ as can be seen from the bottom panel, which displays the differences in all three solution components as a function of time.

making the unaccounted contributions from $g(t)$ as small as possible. Both tasks can be addressed by clever combinations of mechanistic models with observational data. Associated computational techniques are often referred to as *data assimilation* in the geosciences and *filtering/smoothing* in the engineering com-

munity. Throughout this book we will primarily use the term data assimilation, which, broadly speaking, covers the task of combining mechanistic models with partial observations in order to produce skillful forecasts.

To give a flavour of what is to come in Part II of this book, we present an application of the method of least squares to the state estimation of a mechanistic model (1.18). Let us assume that observations $y_{\text{obs}}(\mathfrak{t}_k)$ are available at time instances

$$\mathfrak{t}_k = k\,\Delta t_{\text{out}}, \qquad k = 1, \ldots, N_{\text{A}},$$

where $\Delta t_{\text{out}} = N_{\text{out}}\delta t$ for given integer $N_{\text{out}} \geq 1$. Starting from the initial condition z^0 at $t = 0$, kN_{out} applications of (1.18) produces a model solution $z_{\text{model}}(\mathfrak{t}_k)$ and a simulated observation $y_{\text{model}}(\mathfrak{t}_k) = h(z_{\text{model}}(\mathfrak{t}_k))$, which can be compared to the observed value $y_{\text{obs}}(\mathfrak{t}_k)$. The differences between simulated and true observations is measured in a residual

$$r_k = y_{\text{model}}(\mathfrak{t}_k) - y_{\text{obs}}(\mathfrak{t}_k) = h(z_{\text{model}}(\mathfrak{t}_k)) - y_{\text{obs}}(\mathfrak{t}_k), \qquad k = 1, \ldots, N_{\text{A}}.$$

The residual implicitly depends on the model initial condition z^0, since this changes the entire model trajectory and therefore the simulated observations $y_{\text{model}}(\mathfrak{t}_k)$. To simplify the discussion, we will assume that the forward operator h is linear and is represented by a row vector $H \in \mathbb{R}^{1\times N_z}$, *i.e.*,

$$h(z) = Hz.$$

Adopting the method of least squares, we seek the initial condition z^0 that minimises the residual sum

$$L(z^0) = \frac{1}{2}\sum_{k=1}^{N_{\text{A}}} r_k^2. \tag{1.20}$$

We denote a minimiser by z_*^0, and recall from elementary calculus that z_*^0 has to be a critical point of L to be a candidate for a minimum, *i.e.*, the gradient $\nabla_{z^0} L(z^0) \in \mathbb{R}^{N_z}$ has to vanish at $z^0 = z_*^0$. In contrast with the linear least square method considered previously, we now must solve systems of nonlinear equations to find critical points of L. These critical points may correspond to local minima or even maxima. The main complication arises from the nonlinear dependence of $z_{\text{model}}(\mathfrak{t}_k)$ on z^0. In order to make this dependence explicit, we introduce the map ψ as a shorthand for the N_{out}-fold application of (1.18), *i.e.* if we define

$$z_{\text{model}}(\mathfrak{t}_{k+1}) = \psi(z_{\text{model}}(\mathfrak{t}_k)), \quad k \geq 0,$$

then

$$z_{\text{model}}(\mathfrak{t}_k) = \psi^k(z^0) := \underbrace{\psi(\psi(\cdots\psi(z^0)))}_{k\text{-fold application of } \psi}. \tag{1.21}$$

Computing the gradient of $L(z^0)$ requires the Jacobian matrix of first derivatives of $z_{\text{model}}(\mathfrak{t}_k) \in \mathbb{R}^{N_z}$ with respect to the initial condition $z^0 \in \mathbb{R}^{N_z}$, given by

$$Dz_{\text{model}}(\mathfrak{t}_k) := D\psi^k(z^0) \in \mathbb{R}^{N_z\times N_z}.$$

The Jacobian can be computed from (1.21) directly or using the following recursive approach. First we note that a single application of (1.18) leads to

$$Dz^1 := D(z^0 + \delta t f(z^0)) = I + \delta t Df(z^0),$$

since $Dz^0 = I$ and $Df(z) \in \mathbb{R}^{N_z \times N_z}$ denotes the Jacobian matrix of partial derivatives of $f(z)$. The calculation of Dz^2 can now be decomposed into

$$Dz^2 := D(z^1 + \delta t f(z^1)) = (I + \delta t Df(z^1))\, Dz^1,$$

using the chain rule of differentiation. More generally, one finds the recursion

$$Dz^{n+1} = (I + \delta t Df(z^n))\, Dz^n \tag{1.22}$$

for $n \geq 0$ with Dz^0 equal to the identity matrix. Upon setting $n = kN_{\text{out}}$, we obtain the desired expression for the Jacobian $D\psi^k(z^0)$ and the gradient of L is given by

$$\nabla_{z^0} L(z^0) = \sum_{k=1}^{N_{\text{A}}} (D\psi^k(z^0))^{\text{T}} H^{\text{T}} r_k. \tag{1.23}$$

The minimiser z_*^0 must satisfy

$$\nabla_{z^0} L(z_*^0) = 0.$$

Later in this book we will show that an explicit calculation of the Jacobian $Dz_{\text{model}}(\mathfrak{t}_k)$ *via* the recursion (1.22) is not necessary for determining (1.23). We emphasise again that, in contrast with the linear method of least squares, the existence and uniqueness of a critical point of L is often not guaranteed *a priori*. In addition, critical points may correspond to local maxima or saddle points instead of minima.

A (local) minimum of L can be searched for by the *gradient* or *steepest decent method*, which is an iteration of the form

$$z^{(l+1)} = z^{(l)} - \alpha \nabla_{z^0} L(z^{(l)}) \tag{1.24}$$

for $l \geq 0$ and an appropriate initial guess $z^{(0)}$. The coefficient $\alpha > 0$ needs to be chosen sufficiently small in order to guarantee

$$L(z^{(l+1)}) \leq L(z^{(l)})$$

throughout the iteration process. More refined gradient methods would choose the coefficient α adaptively (see Nocedal & Wright (2006), for example).

Let us assume that we have obtained a reasonable estimate for the initial state z^0 of our mechanistic model and that a series of N_{A} observations within the assimilation window become available at $\mathfrak{t}_1, \ldots, \mathfrak{t}_{N_{\text{A}}}$. Then we can iterate (1.24) with starting value $z^{(0)} = z^0$ until

$$\|\nabla_{z^0} L(z^{(l_*)})\| \leq \varepsilon, \tag{1.25}$$

where $\varepsilon > 0$ is a desired tolerance. The resulting $z_*^0 = z^{(l_*)}$ is often called the

analysis at $t = 0$ and we have completed what is often called a *variational data assimilation* cycle.

Once the analysis z_*^0 has been obtained, we can use the mechanistic model (1.18) with adjusted initial condition $z^0 = z_*^0$ to obtain *forecasts* $z_{\text{model}}(t)$ for $t > \mathfrak{t}_{N_\text{A}}$. In due course, a new sequence of observations $y_{\text{obs}}(\mathfrak{t}_{N_\text{A}+k})$, $k = 1, \ldots, N_\text{A}$, becomes available. At this point a new data assimilation cycle can be initiated. More specifically, we can repeat the above nonlinear least square method by minimising the cost functional $L(z^0)$ with residuals r_k, $k = 1, \ldots, N_\text{A}$, now given by

$$r_k = H\psi^k(z^0) - y_{\text{obs}}(\mathfrak{t}_{N_\text{A}+k})$$

and starting value $z^{(0)} = z_{\text{model}}(\mathfrak{t}_{N_\text{A}})$ in (1.24).[3] The information from the previous sequence of observations feeds in *via* the choice of initial condition for the steepest descent calculation, which may select a particular local minimum; this may also speed up the calculation by starting closer to the minimum. It is often also desirable to make better use of this information by including a penalty term in the functional that becomes large if z^0 gets too far from this initial guess, encoding our belief that the previous data assimilation cycle gave us a good guess for the current system state. This presents a difficulty: we must then decide how much weight in the functional to give the previous forecast relative to the new observational data. We leave this problem for now, but it is a central topic for the rest of the book.

In contrast with the forecasted values $z_{\text{model}}(t)$, $t > \mathfrak{t}_{N_\text{A}}$, which do not make use of the observations $y_{\text{obs}}(\mathfrak{t}_{N_\text{A}+k})$, $k \geq 1$, the minimiser z_*^0 of L provides now an improved approximation $z_{\text{model}}(t)$, called the *analysis*, using the mechanistic model (1.18) with adjusted initial condition $z_{\text{model}}(\mathfrak{t}_{N_\text{A}}) = z_*^0$. In order to distinguish the forecast from the analysis we introduce the notation $z^{\text{f}}_{\text{model}}(\mathfrak{t}_k)$ for the forecast at time $\mathfrak{t}_k$, $k = N_\text{A} + 1, \ldots, 2N_\text{A}$, and $z^{\text{a}}_{\text{model}}(\mathfrak{t}_k)$ for the analysis arsing from the adjusted initial condition at $\mathfrak{t}_{N_\text{A}}$.

Once time t is increased beyond $t = \mathfrak{t}_{2N_\text{A}}$ the analysis $z^{\text{a}}_{\text{model}}(t)$ becomes a forecast and, as soon as all necessary observations have become available, $z^{\text{a}}_{\text{model}}(\mathfrak{t}_{2N_\text{A}})$ is taken as the starting value $z^{(0)}$ for the next assimilation cycle covering the interval $[\mathfrak{t}_{2N_\text{A}}, \mathfrak{t}_{3N_\text{A}}]$. The process of producing forecasts with the mechanistic model and correcting them by assimilating available observations over finite-time intervals can now be repeated as often as desired, as illustrated in Figure 1.10.

Each data assimilation cycle effectively leads to a nudging of the model output towards observations. In other words, the sequence of data assimilation cycles should ideally synchronise the model (1.18) with the physical process via partial observations and corresponding adjustments in model forecasts $z^{\text{f}}_{\text{model}}(\mathfrak{t}_{mN_\text{A}})$,

[3] The simplification of always minimising with respect to z^0 and to only change the observations in the definition of the residuals r_k is possible since our mechanistic model (1.18) is assumed to be time-independent.

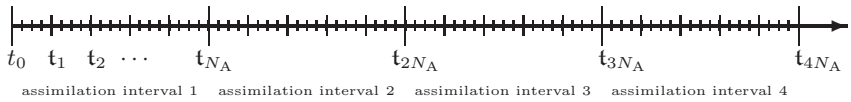

Figure 1.10 Diagram illustrating model timesteps, observation times, and assimilation intervals. At the end of each assimilation window, a new trajectory is computed that uses the observations made during that window.

$m \geq 1$. The most recently adjusted model state is then used as an initial condition for model-based forecasts further into the future.

We now demonstrate this synchronising effect with our "mechanistic" Lorenz-63 model from Example 1.7.

Example 1.8 We implement the nonlinear least square method for the Lorenz-63 model already investigated in Example 1.7. Recall that the model output deviates from the surrogate physical process after a relatively short time-interval even if both the mechanistic model (1.18) and the reference model (1.2) are started from identical initial conditions at $\mathrm{t}_0 = 0$. We now consider sequences of $N_\mathrm{A} = 5$ observations with observation interval $\Delta t_\mathrm{out} = 0.05$ in order to adjust the model's initial states over each data assimilation window $[\mathrm{t}_{mN_\mathrm{A}}, \mathrm{t}_{(m+1)N_\mathrm{A}}]$ with $m = 0, 2, \ldots, 39$. See Figure 1.11 for a comparison between the reference trajectory $z_\mathrm{ref}(t)$ and the analysis $z^a_\mathrm{model}(t)$ over all 40 assimilation cycles, and Figure 1.12 for a zoomed region displaying the difference between model forecasts $z^\mathrm{f}_\mathrm{model}(t)$ and their analysis $z^a_\mathrm{model}(t)$. The nonlinear method of least squares is implemented with step-length $\alpha = 0.05$ in the gradient method (1.24) and $\varepsilon = 0.01$ in (1.25). This small value of α is necessary in order to avoid a divergence of (1.24). More efficient minimisation methods could be implemented but are outside the scope of this book.

In practical applications, such as weather forecasting, it is desirable to obtain *a priori* estimates of the likelihood of an analysis being within a certain range of the (generally not explicitly available) true system state. This gives an indication of how seriously to take the forecast; this is crucial when using forecasts in decision-making and planning. We display a histogram of the resulting differences between the reference solution $z_\mathrm{ref}(t)$ and the analysis $z^a_\mathrm{model}(t)$ in Figure 1.13. Note, for example, that the errors in the z-component have a much broader distribution than those in the x-component. More abstractly, we will view histograms, such as the ones displayed in Figure 1.13, as resulting from finite samples of underlying *random variables* with generally unknown distribution. It is a task of *uncertainty quantification* to provide as much information about these random variables as possible.

We have already mentioned that the analysis can be used to generate forecasts over time intervals where observations have not yet been obtained. In order to illustrate this aspect of data assimilation we use the analysis $z^\mathrm{a}_\mathrm{model}(\mathrm{t})$ at time

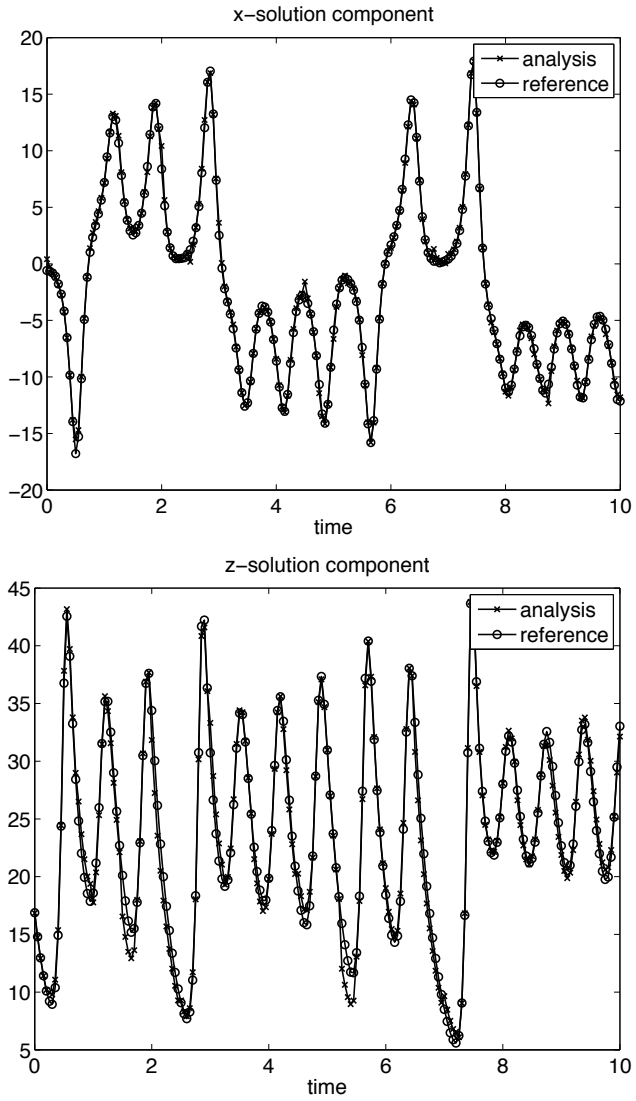

Figure 1.11 We display the results from 40 data assimilation cycles each over a window of length $5\,\Delta t_{\text{out}} = 0.25$. Only the x-variable is observed in intervals of $\Delta t_{\text{out}} = 0.05$. The synchronising effect of the nonlinear least square approach can be clearly seen both in the x variable and the unobserved z variable, while the model output without adjustments from the data assimilation cycles loses track of the underlying physical process at about $t = 2$. Compare Figure 1.9.

$t = 10$ as the initial condition for our model (1.18) at $t = 10$. We then run this model over the time interval [10,15] in order to produce a forecast which can be compared to the reference solution $z_{\text{ref}}(t)$ of our surrogate physical process

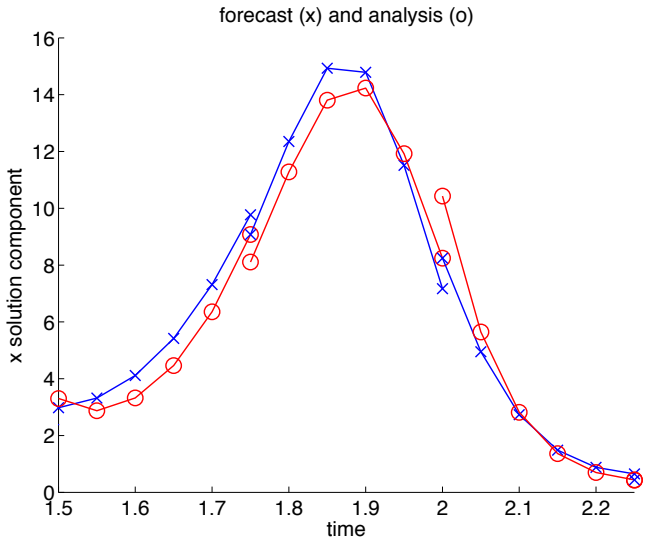

Figure 1.12 We display the forecast (which does not take the observations into account) and the analysis (which has assimilated the observations) for three data assimilation cycles. The forecast is in blue and marked with crosses, and the analysis is in red and marked with circles. At the beginning of each assimilation window, the previous forecast terminates, the most recent analysis turns into a forecast and eventually provides the starting value for the next assimilation cycle. This can be best seen at $t = 1.75$, where the upper cross marks the last step of the forecast starting at $t = 1.5$. The circles between $t = 1.5$ and $t = 1.75$ represent the subsequent analysis using the data available from that interval. The next forecast (crosses) starts from the last analysis (circle) at $t = 1.75$. This forecast window ranges from $t = 1.75$ to $t = 2.0$. The process then repeats, and this new forecast is modified by the data available from the interval $[1.75, 2.0]$. This data assimilation step provides the second, lower circle at $t = 2.0$ as well as the starting value of a new forecast over the interval $[2.0, 2.25]$.

over the same time interval. The result is displayed in Figure 1.14, where it can be seen that the forecast stays close to the reference solution up to time $t \approx 12$. It is a task of uncertainty quantification to quantify the actual forecast uncertainties without explicit knowledge of the reference solution. In addition to analysis errors, forecast errors will be also treated as random variables. We will learn about computational methods for estimating their distributions later in this book.

We conclude this example by emphasising that the nonlinear least square method is sensitive to the length of the assimilation window. If the window is chosen too large, then the data assimilation procedure leads to a divergence of the gradient method (1.24) due to the increasingly nonlinear behaviour of the functional L. We also found that a shorter window of $2\Delta t_{\text{out}}$ (*i.e.* two observations per assimilation cycle) leads to results similar to those displayed in Figure 1.11 for 5 observations per assimilation cycle. This is surprising, since with $N_{\text{A}} = 2$ we

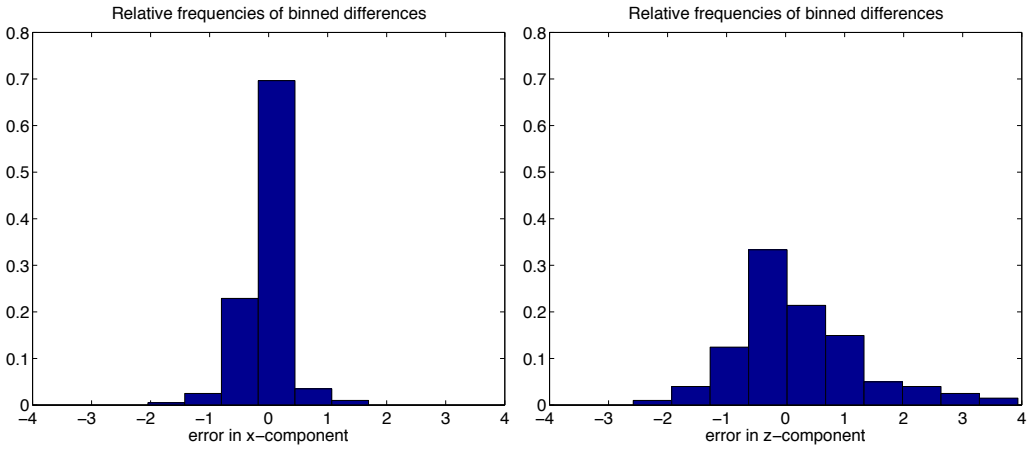

Figure 1.13 Relative frequencies of binned differences between the reference solution and the analysis, in both x and z. It is tempting to view these relative frequencies as arising from finite samples of an underlying random variable with unknown specifications. We could then discuss the probability of an analysis falling within a certain range of the (generally unavailable explicitly) true system state. It is a task of uncertainty quantification to characterise such probabilities.

cannot expect that L has a unique (local) minimum. In particular, the computed minimiser z_*^0 will depend on the initial guess for the steepest decent method. As we will see in later chapters, such a dependence is not necessarily a disadvantage.

The previous example has demonstrated the effectiveness of using observations to estimate the state of a mechanistic model, then using the model with adjusted initial conditions to generate forecasts. The nonlinear method of least squares provides us with the first example of a data assimilation algorithm, which also goes under the name of *variational data assimilation* or *maximum likelihood estimation*. While the results are encouraging, we will learn about even more sophisticated data assimilation methods later in this book. These methods rely on a probabilistic interpretation of model forecasts as well as measurement errors. The necessary mathematical tools will be provided in subsequent chapters. In particular, we will need to introduce the concept of a *random variable*, together with methods for performing computer experiments with random variables (*Monte Carlo methods*). We will then apply these tools to mechanistic models of type (1.18) to describe *stochastic processes* and *forecast uncertainties*. We also need to introduce *Bayesian inference* as a probabilistic framework for inferring information on random variables from (partial) observations and available prior knowledge about the random variable under consideration. Once these mathematical foundations have been established, the second part of this book on data assimilation algorithms can be entered.

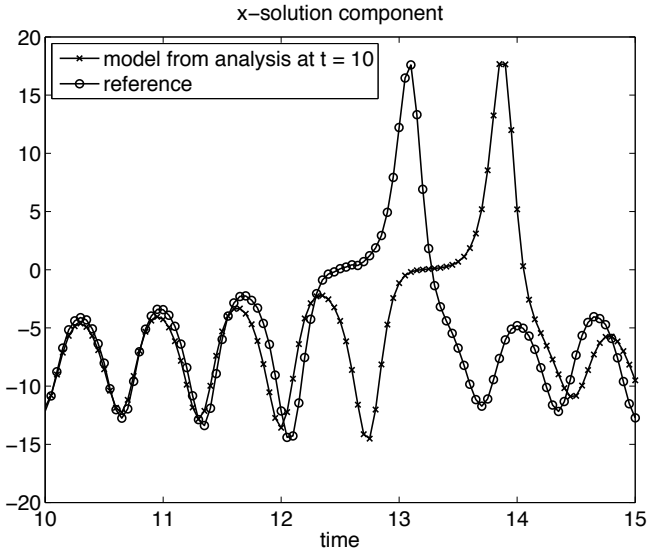

Figure 1.14 Forecast started from the analysis at time $t = 10$ and reference solution from the surrogate physical process. It can be seen that the forecast stays close to the reference solution for about 2 time units after which its starts diverging due to errors in the analysis and model errors.

We end this introductory chapter with some general comments on the process of building mathematical models for making predictions and the role of data assimilation within this process. We can use well-established theories, such as Newton's laws of motion, to build mathematical models in the form of evolution equations/dynamical systems as encountered in this chapter in Equation (1.18). These models can be discretised and programmed on a computer, allowing mathematical experiments to be performed. The analysis of such computational models falls, broadly speaking, into the world of *applied computational mathematics*. When making predictions, this is one side of the coin; the other side is the world of measured data and their mathematical structure. This area is traditionally investigated from a mathematical perspective by *statisticians*. While the two sides of the same coin have largely been developed independently over the years, there is an increasing awareness that they belong together and that maximal benefit can be gained by combining them within the framework of *data assimilation*. Data assimilation can be used for state estimation, for adapting models through parameter estimation, and for intercomparison of models. The Bayesian formalism has emerged as a particularly fruitful approach to data assimilation and is primarily the approach that we will follow in this book. See Figure 1.15 for a graphical illustration of this discussion.

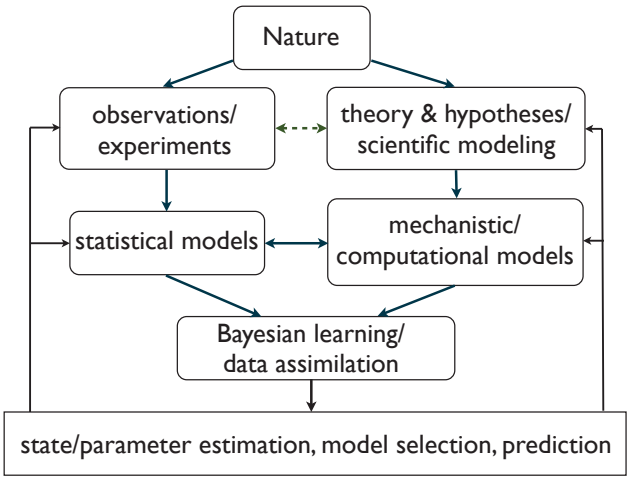

Figure 1.15 A schematic presentation of the complex process of building and using mathematical models. An essential building block is data assimilation which is a mathematical technique for combining mechanistic models with statistical models based on data. Here a statistical model refers to a set of probability distributions describing observed data, while a mechanistic model consists of evolution equations which give rise to deterministic and/or stochastic processes. Mechanistic models depend on parameters (including the state of the model), which we treat as random variables. Finally, a computational model is typically an algorithm that allows us to produce process realisations of the mechanistic model. Among the possible outcomes of data assimilation are improved predictions through parameter and state estimation as well as model selection.

Problems

1.1 Implement the numerical model from Example 1.1 and store the resulting reference trajectory in time intervals of $\Delta t_{\text{out}} = 0.05$ in a file for later use in other examples. Do not store the system state from every single timestep as this becomes very inefficient, even for low dimensional problems; it is much better to overwrite the state vector on each timestep, and take a copy of the vector when you need to store it. The resulting data set should be stored in a matrix of size 3×4001. Do not forget to replace (1.6) by (1.27) and check that your numerical results reproduce the model attractor from Figure 1.1.

1.2 Implement the numerical observation process as defined in Example 1.2 using the reference trajectory generated in Problem 1.1. Store the numerically generated observation values $y_{\text{obs}}(\mathfrak{t}_k)$ for $k = 1, \ldots, N_{\text{obs}} = 4000$, in a file for later use. Investigate the difference between using the two recursions (1.28) and (1.9) for generating measurement errors. Hint: You might obtain a trajectory different from the one displayed in Figure 1.3. Differences can arise even for mathematically identical implementations due to round-off errors.

1.3 Follow Example 1.4 and use linear extrapolation in order to produce fore-

casts from the observations produced in Problem 1.2. Compute the time averaged RMSE and compare to the values produced in Problem 1.4.

1.4 Implement the method of least squares in the setting laid out in Example 1.6. Compare your coefficients a_l, $l = 1, \ldots, 4$, to the one stated in Example 1.6. Also, compare the resulting time averaged RMSEs.

1.5 Implement the mechanistic model for our surrogate physical process as described in Example 1.7. Use (1.5) as initial conditions at $t_0 = 0$. Your results should reproduce the phase portrait displayed in Figure 1.8. Now compare the model trajectory to the reference trajectory $z_{\mathrm{ref}}(t)$ computed in Exercise 1.1. Next, vary the initial conditions $(\mathrm{x}_0, \mathrm{y}_0, \mathrm{z}_0)$ for the mechanistic model by adding arbitrary perturbations $(\delta_x, \delta_y, \delta_y) \in [-1, 1]^3$ and study the impact of these perturbations on the forecast quality with respect to the reference trajectory $z_{\mathrm{ref}}(t)$.

1.4 Guide to Literature

A mind-provoking introduction to many of the topics raised in this chapter can be found in Smith (2007*b*). Another book from the same series by Hand (2008) discusses some of the data related issues in more detail. The Lorenz-63 model was introduced by Lorenz (1963) and provides a widely studied chaotic dynamical system. Polynomial interpolation and the method of least squares are covered in Süli & Mayers (2006). A broad overview of minimisation techniques including the nonlinear method of least squares is given in Nocedal & Wright (2006). The importance of uncertainty quantification and data assimilation for numerical weather forecasting is demonstrated in Kalnay (2002). In fact, numerical weather forecasting provides an ideal motivation as well as an application area for most of the material covered in this book.

1.5 Appendix: Numerical implementation of tent map iteration

In this appendix, we discuss the implementation on a computer of the tent map iteration

$$g^{n+1} = \begin{cases} 2g^n & \text{if } g^n \in [0, 1/2), \\ 2(g^n - 1) & \text{otherwise,} \end{cases} \tag{1.26}$$

for given initial $g^0 \in [0, 1]$ and $n \geq 0$. We consider the case where this iteration is approximated by a binary floating point representation of the non-negative real numbers g^n in the form of

$$g^n = m \times 2^l,$$

where

$$m = 0.m_1 m_2 m_3 \cdots ,$$

is called the *mantissa* and $m_i \in \{0,1\}$ are the mantissa *bits*. The integer l is called the *exponent* of the representation. For efficient storage, the exponent l is chosen such that $m_1 = 1$ unless $m = 0$. If the mantissa has only finitely many non-zero bits, then i^* denotes the largest i such that $m_i = 1$. We call i^* the *mantissa length*. As an example, consider

$$g^n = 0.1101 \times 2^2,$$

with mantissa length $i^* = 4$, which corresponds to

$$g^n = 1 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} = 3.25,$$

in our standard decimal representation. We also find that $g^n = 1$ corresponds to 0.1×2^1 in our binary representation. Furthermore, $g^n \in [0, 1/2)$ implies that the associated exponent l satisfies $l \leq -1$. Similarly, we find that $g^n \in [0,1]$ implies $l \leq 1$.

Now consider the tent map iteration written in this representation. Straightforwardly, $2g^n$ leaves the mantissa of g^n unaltered while its exponent is increased by one. Understanding the impact of $2 - 2g^n$ on $g^n \in [1/2, 1)$ is more delicate. We obtain

$$\begin{aligned} 0.1 \times 2^2 - 0.1m_2m_3 \cdots \times 2^1 &= (0.1 - 0.01m_2m_3 \cdots) \times 2^2 \\ &= 0.00\overline{m}_2\overline{m}_3 \cdots \times 2^2 \\ &= 0.\overline{m}_2\overline{m}_3 \cdots \times 2^0, \end{aligned}$$

where $\overline{m}_i = m_i + 1$ modulo 2 for all $i = 1, \ldots, i^* - 1$. If i^* is finite, then we obtain $\overline{m}_{i^*} = m_{i^*} = 1$. In this case, the mantissa length of g^{n+1} gets reduced by at least one (it is exactly one in the case in which $\overline{m}_2 = 1$). Consequently, if the initial g^0 is chosen such that its mantissa m has finite length, then the tent map will ultimately lead to $g^n = 0$ after sufficiently many iterations of the tent map. Of course, a number with infinite mantissa length will lead to zero for any number of tent map iterations.

Since digital computers rely on a binary representation of real numbers with a mantissa of necessarily finite length, any computer implementation of the tent map iteration will ultimately result in $g^n = 0$, after sufficiently many iterations. This is in stark contrast to the "true" tent map, which generically produces infinite, non-repetitive sequences $\{g_n\}_{n=0}^{\infty}$. This example should serve as a warning not to identify a mathematical model with a computer implementation of it.

The following modification mimics the desired asymptotic behaviour of the tent map iteration when implemented in finite floating point arithmetic:

$$g^{n+1} = \begin{cases} 1.99999 g^n & \text{if } g^n \in [0, 1/2), \\ 1.99999(g^n - 1) & \text{otherwise.} \end{cases}$$

Similarly, the following modifications were applied to the iterations (1.6) and

(1.9), respectively:

$$g_i^{n+1} = \begin{cases} 1.99999 g_i^n + a/2 & \text{if } g_i^n \in [-a/2, 0), \\ -1.99999 g_i^n + a/2 & \text{otherwise,} \end{cases} \tag{1.27}$$

$$\xi_{k+1} = \begin{cases} 1.99999 \xi_k + a/2 & \text{if } \xi_k \in [-a/2, 0), \\ -1.99999 \xi_k + a/2 & \text{otherwise.} \end{cases} \tag{1.28}$$

Part I

Quantifying Uncertainty