21 Data Assimilation Methods

Most of the data-driven techniques presented in this book were applied to systems where the underlying governing equations were prescribed. However, in the DMD method (or in the equation-free method), no governing equations were required to extract meaningful information about the dynamics of the complex system under consideration. The method of *data assimilation* is a hybrid method that uses both data measurements collected in time about the system in conjunction with a prescribed set of governing equations. The fact is, both the measurements and simulations are in practice heavily influenced by uncertainty and/or noise fluctuations. Thus neither the experiment or theoretical model can be fully trusted. However, combining the two so that the experimental measurements helps inform the model and vice-versa can greatly improve the predictive powers of the model, or analysis of the state of the system [83, 84, 85, 86]. Thus data is assimilated into the model predictions and hence the name. Data assimilation is potentially one of the most useful data-driven modeling techniques as we are rarely without some underlying governing equations or without experimental measurements. And to make optimal use of both, data assimilation techniques are ideal.

21.1 Theory of Data Assimilation

To begin thinking about data assimilation, we will first consider a given complex system and its underlying dynamical evolution. Specifically, let us begin by assuming that the system under consideration has the following evolution equation

$$\frac{d\mathbf{y}}{dt} = f(t, \mathbf{y}) \tag{21.1.1}$$

with the initial state of the system given by

$$\mathbf{y}(0) = \mathbf{y}_0 \,. \tag{21.1.2}$$

Of course, techniques for solving such a system have been considered extensively throughout this book. Indeed, provided that $f(t, \mathbf{y})$ is well behaved, i.e. continuous and differentiable, for instance, then a solution to the prescribed problem can be solved for uniquely. In fact, for an N-dimensional vector $\mathbf{y}$, N initial conditions are prescribed by $\mathbf{y}_0$ so that the number of unknowns and constraints match.

Up to this point, we have ignored and/or denied a simple and obvious truth regarding the evolution equation (21.1.1) and its initial conditions (21.1.2). Specifically, we are assuming that we can perfectly prescribe both! In practice, this is surely an impossible task. And for strongly nonlinear systems (21.1.1), even small changes in the initial conditions and/or slight noise perturbations to the system can lead to large changes in the dynamical behavior, stability and

predictability of (21.1.1). To be more precise then about our formulation of the true system, (21.1.1) and (21.1.2) should be modified to

$$\frac{d\mathbf{y}}{dt} = f(t, \mathbf{y}) + \mathbf{q}_1(t) \tag{21.1.3}$$

with the initial conditions

$$\mathbf{y}(0) = \mathbf{y}_0 + \mathbf{q}_2 \,. \tag{21.1.4}$$

where $\mathbf{q}_1$ represents unknown model errors due to either noise fluctuations in the system or perhaps truncation of higher-order effects in the system that are thought to be negligible. Similarly, the vector $\mathbf{q}_2$ represents the error in the prescribed initial conditions either because we cannot accurately measure them in practice or prescribe them in a realistic physical system.

Even in the presence of the perturbations $\mathbf{q}_1$ and $\mathbf{q}_2$, the system of equations (21.1.3) and (21.1.4) remains well-posed with a unique solution specifying the dynamics in time. However, for a strongly nonlinear system, the presence of $\mathbf{q}_1$ and $\mathbf{q}_2$ make it virtually impossible to use the governing equations (21.1.3) and the initial conditions (21.1.4) for an accurate prediction of the future state of the system. This is largely due to the concept of *sensitivity to initial conditions* that is displayed in many complex systems. Of course, this then gives rise to the following fundamental question: Is modeling a worthwhile exercise if it cannot predict the future state of a realistic system? In practice, great engineering is often about eliminating large unknowns in a system by essentially suppressing, as much as possible, the vectors $\mathbf{q}_1$ and $\mathbf{q}_2$. But for highly complex systems, such as climate modeling and weather prediction, the system simply cannot be engineered and one must find effective strategies for dealing with the inherent effect of $\mathbf{q}_1$ and $\mathbf{q}_2$.

Data assimilation is a technique that attempts to mitigate the problems associated with having $\mathbf{q}_1$ and $\mathbf{q}_2$ present in a given system. As the name suggests, the idea is to assimilate experimental measurements directly into the theoretical model in order to inform the dynamics. Thus a set of measurements are taken so that

$$g(t, \mathbf{y}) + \mathbf{q}_3 = 0 \tag{21.1.5}$$

where $g(t, \mathbf{y})$ is a certain set of measurements, let's say M of them, on some quantities related to the state vector $\mathbf{y}$, and the vector $\mathbf{q}_3$ is the error measurement associated with the data collection process.

In principle, it is a great idea to incorporate real experimental measurements into the modeling process. In practice, the addition of (21.1.5) now makes for an overdetermined system (N unknowns and $N + M$ constraints) for $\mathbf{y}$ for which no solution exists in general. Thus the tradeoff of using the experimental data is that we went from a well-posed system with a unique solution to an overdetermined system with no general solution.

To deal with the overdetermined system, the following *quadratic form* is introduced:

$$J(\mathbf{y}) = \int_0^T \int_0^T \mathbf{q}_1^T(t_1)\mathbf{W}_1\mathbf{q}_1(t_2)dt_1dt_2 + \mathbf{q}_2^T\mathbf{W}_2\mathbf{q}_2 + \mathbf{q}_3^T\mathbf{W}_3\mathbf{q}_3 \qquad (21.1.6)$$

where the error vectors $\mathbf{q}_j$ are all directly included in the quadratic form. The matrices $\mathbf{W}_j$ are the inverse of the error covariance for the model, initial conditions and measurements respectively. The introduction of such a quadratic form is motivated by one primary purpose: optimization. In particular, one potential solution to the overdetermined system is to find the solution that minimizes the weighted error, weighted with respect to the model, initial condition and measurement error, as given by the quadratic form $J(\mathbf{y})$. Since it is a quadratic form, standard convex optimization methods can be directly applied to find a solution. The least-square error model defined by $J(\mathbf{y})$ is only one potential choice. However, this choice is particularly attractive when considering Gaussian statistics for the error vectors. In this case, minimizing $J(\mathbf{y})$ is equivalent to maximizing the probability density function $P(\mathbf{y}) = C\exp[-J(\mathbf{y})]$. Thus the minimum of (21.1.6) is also the maximum-likelihood estimate.

Thus to summarize the data assimilation method, we consider a theoretical model under the influence of some error (21.1.3) subject to initial conditions which are also subject to error (21.1.4). A number of experimental measurements (21.1.5), also subject to error, are then made to inform the model. The combined errors of the resulting overdetermined system is cast as a quadratic form for which convex optimization techniques can be used to find the best fit solution (in the L^2 sense). As a result, the experimental measurements are assimilated with the model predictions to generate better predictions of the dynamical behavior in time. The development of these key ideas will be carried forward in the next two sections.

Data Assimilation for a Single Random Variable

To illustrate the ideas of data assimilation, the simplest toy example will be considered (See, for example, the nice arguments by Holton and Hakim [82] in the context of atmospheric sciences). Specifically, consider a model that generates some prediction about the state of the system, thus effectively $\mathbf{q}_1$ and $\mathbf{q}_2$ would be combined in this model prediction process. Also, consider some experimental measurements on the systems that are also subject to error. Thus for the variable x, there are two predictions about its true value: one from the model and one from experiment. The idea is to combine these two measurements to arrive at a better prediction of the true value of x.

To begin this calculations, consider the following conditional probability statement from Bayes formula (See Sec. 12.2):

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)} \qquad (21.1.7)$$

where $p(x)$ represents the probability density for predicting the variable x. Here, y will represent the observation (experimental assimilation) that will be made. Thus $p(x|y)$ is the probability of finding x conditioned on having measured y. At this point, we will not concern ourselves with $p(y)$ since it will simply represent a scaling factor for (21.1.7). The probability density function $p(y|x)$ is a *likelihood function* since it is a function of the variable x.

As is the case with almost all problems in probability theory, great simplifications can be made by assuming Gaussian distributed random variables. In fact, our treatment of the data assimilation problem for higher dimensional problems relies explicitly on this assumption in order to derive something tractable. It also helps simplify the current one variable problem if this assumption is made. Thus consider the following probability density distributions

$$p(y|x) = c_1 \exp\left[-\frac{1}{2}\left(\frac{y-x}{\sigma_y}\right)^2\right] \tag{21.1.8}$$

$$p(x) = c_2 \exp\left[-\frac{1}{2}\left(\frac{x-x_0}{\sigma_0}\right)^2\right] \tag{21.1.9}$$

where σ_y is the error variance for the observation, and x_0, σ_0 are the predicted model mean and its associated error variance. The constants c_1 and c_2 are normalization constants ensuring that probability density functions integrate to unity. Thus the errors, both in measurement (σ_y) and theory (σ_0), are characterized by the variance parameters. Note that without any data assimilation, the model would predict the value x_0. Data assimilation attempts to give a correction to this value using the measurement data.

Using the conditional probability statement, which integrates in information about the observation y, along with the assumed Gaussian probability distributions (21.1.9) yields the following prediction (what is x given observation y) for the state of the system

$$p(x|y) = c_3 \exp\left[-\frac{1}{2}\left(\frac{y-x}{\sigma_y}\right)^2\right] \exp\left[-\frac{1}{2}\left(\frac{x-x_0}{\sigma_0}\right)^2\right] \tag{21.1.10}$$

where c_3 is another constant used for convenience.

Our goal now is to construct a quadratic form like (21.1.6) for this problem in order to determine how to modify x from its default prediction value of x_0 in light of our observation y. A very simple quadratic form to construct from the Gaussian distributions is as follows

$$J(x) = -\log[p(x|y)] + \log(c_3) = \frac{1}{2}\left(\frac{y-x}{\sigma_y}\right)^2 + \frac{1}{2}\left(\frac{x-x_0}{\sigma_0}\right)^2. \tag{21.1.11}$$

But this quadratic form can be easily minimized as it is simply a parabola in one dimension, i.e. optimization can be performed in closed form unlike the

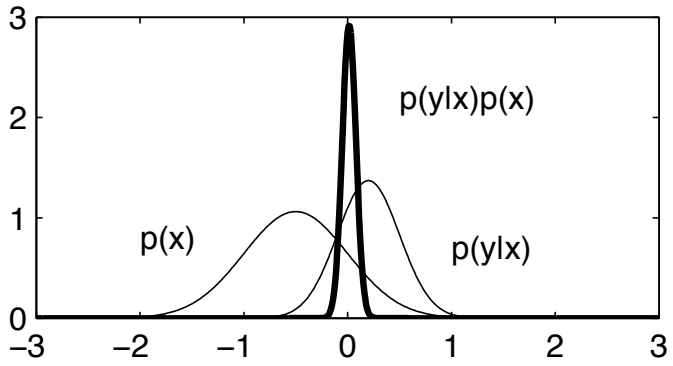

Figure 238: Distributions of the model (left Gaussian, $p(x)$), the observation (right Gaussian, $p(y|x)$), and the data assimilated distribution $p(\bar{x}) = p(y|x)p(x)$. Note that the error variance of the assimilated distribution is much narrower than either the model or observational data as shown in (21.1.13).

general formulation (21.1.6). To find its minimum, $dJ(\bar{x})/dx = 0$ is computed. This yields the following value for $\bar{x}$ at the minimum

$$\bar{x} = \left(\frac{\sigma_y^2}{\sigma_y^2 + \sigma_0^2}\right) x_0 + \left(\frac{\sigma_0^2}{\sigma_y^2 + \sigma_0^2}\right) y\,. \tag{21.1.12}$$

Thus $\bar{x}$, which is a weighted linear superposition of the model prediction x_0 and the observation y, is the new and improved prediction for the outcome of x given the observation y. Such is change of value when using the conditional probability argument.

Equation (21.1.12) has some very intuitive features. First and foremost: if there is no error in the experimental observation, then $\sigma_y = 0$ and $\bar{x} = y$. This is a statement of self-consistency essentially. It would be very bad if the data assimilation method failed to choose the observational data value if it was error free. The error variance for $\bar{x}$ can also be computed to be

$$\bar{\sigma}^2 = \frac{\sigma_0^2}{1 + (\sigma_0^2/\sigma_y^2)} = \frac{\sigma_y^2}{1 + (\sigma_y^2/\sigma_0^2)} < \sigma_0^2, \sigma_y^2\,. \tag{21.1.13}$$

This is also a self-consistency check. Namely, the error in the data assimilation prediction is always better than the error of either the model alone or observation alone. Again, the fact that you are using the combination of model and data should always improve your predictions and error.

Figure 238 shows the three probability densities of interest. In particular, what is shown is the probability density given by the model, $p(x)$, the observation, $p(y|x)$ and the data assimilation, $p(\bar{x}) = p(y|x)p(x)$. For this figure the

following were assumed $x_0 = -0.5$, $\sigma_0 = 0.5$, $y = 0.2$ and $\sigma_y = 0.3$. This allows us to compute the distribution of the assimilated prediction through (21.1.12) and (21.1.13). Note that the variance of the assimilated distribution is quite narrow and shifted strongly towards the observational distribution. This is largely because the observation has a much smaller variance than the model data.

Another way to express these results if by noting that

$$\bar{x} = x_0 + K(y - x_0) \tag{21.1.14}$$

with $\bar{\sigma}^2 = (1 - K)\sigma_0^2$ and where

$$K = \frac{\sigma_0^2}{\sigma_0^2 + \sigma_y^2} \leq 1\,. \tag{21.1.15}$$

The second term in the right hand side of (21.1.14), $K(y - x_0)$, is the so-called *innovation* since it brings in new information (an observation) to the prediction of x. The predicted value of x is a linear combination of its model prediction x_0 and the innovation. If there is no innovation, i.e. no new information, then the prediction remains x_0. The parameter $0 < K < 1$ is the gain weighting factor and is essentially the so-called *Kalman filter* or *Kalman gain*. This will be considered in more detail in the next section.

21.2 Data Assimilation, Sampling and Kalman Filtering

The preceding section outlined the high level view of the data assimilation method and showed how to implement it on the simplest example possible. What is desired now is to develop the theory further for implementation in realistic systems. Of particular note will be the role of the *Kalman Filter*, or as already hinted at in the last section, the method of incorporating *innovation* to the model predictions.

Relating Observations to the State Vector

Before proceeding to derive the Kalman filter for a general vector system, we address the issue of how observations (21.1.5) project onto the state variable $\mathbf{y}$ of the governing equation (21.1.1). For instance, the governing equation (21.1.1) may be the simulation of some underlying PDE system that is solved on a rectangular grid of a prescribed domain. Observations, however, may be made anywhere in the domain and will, in general, not be aligned with the grid used for (21.1.1). In weather prediction, the data is collected at observations points which may be on the coast, on top of mountains and/or metropolitan areas. Certainly the observation points are not aligned on a regular grid. The simulation of a geographic region, however, is most likely accomplished using a given discretization, perhaps in tens of meters to kilometers. The question is how to overlay the irregularly spaced observations onto the regularly spaced state

variables. The simplest thing to do is to use a linear interpolation algorithm to generate values of the state variables at each grid point of the model.

The mapping of the experimental observations can be accomplished mathematically in the following way

$$\mathbf{y}(t) = \mathbf{H}\mathbf{x}(t) + \mathbf{q}_3 \tag{21.2.1}$$

where $\mathbf{y}(t)$ are the observations at a given time t of the state vector $\mathbf{x}$, $\mathbf{H}$ is a matrix that maps the state vector to the observations and $\mathbf{q}_3$ is the observational error. This is a linear version of the more general form (21.1.5). This step is always necessary once a data collection has been applied at a time t.

The One-Dimensional Kalman Filter

The derivation of the projection operator $\mathbf{H}$ in (21.2.1) is necessary for deriving the Kalman Filter. To start the derivation process, we once again return to one-dimensional considerations and a highly idealized version of the dynamics in discrete form. In particular, following Miller [84], we can consider the mapping from a time t_k to a time t_{k+1}. The full dynamics, without approximation, is assumed to be given by

$$x_{k+1} = f(x_k) + q_{k+1} \tag{21.2.2}$$

where x_{k+1} and x_k are the state of the one dimensional system at time t_{k+1} and t_k respectively, and q_{k+1} is a Gaussian white noise sequence. The model approximation to this system is given by

$$x_{0_{k+1}} = f\left(x_{0_k}\right) \tag{21.2.3}$$

where x_{0_k} is the best estimate of the state of the system at time t_k, sometimes called the *analysis*, and $x_{0_{k+1}}$ is the forecast of the dynamics using this estimate at time t_{k+1}.

The error between the truth and the forecast at time t_{k+1} is then given by

$$x_{k+1} - x_{0_{k+1}} = f(x_k) - f\left(x_{0_k}\right) + q_{k+1} \tag{21.2.4}$$

By Taylor expanding $f(x_k)$ around x_{0_k}, the above expression can be written as follows

$$\begin{aligned} x_{k+1} - x_{0_{k+1}} = (x_k - x_{0_k})f'(x_{0_k}) + \frac{1}{2}(x_k - x_{0_k})^2 f''(x_{0_k}) \\ + \frac{1}{6}(x_k - x_{0_k})^3 f'''(x_{0_k}) + \cdots + q_{k+1} \end{aligned} \tag{21.2.5}$$

To find the error variance between the correct solution x_{k+1} and our prediction $x_{0_{k+1}}$, the expectation value of the quantity $(x_{k+1} - x_{0_{k+1}})^2$ is computed. Denoting this as $E[(x_{k+1} - x_{0_{k+1}})^2]$, we find by squaring the above expression and taking the expectation

$$E[(x_{k+1} - x_{0_{k+1}})^2] = E[(x_k - x_{0_k})^2](f'(x_{0_k}))^2 + \text{h.o.t} + E[q_{k+1}^2] \tag{21.2.6}$$

where h.o.t. denotes all higher-order moment terms, i.e. the skewness and kurtosis, for instance, of the error. Keeping these terms represents a closure problem for the error. However, as a first approximation, it is often the case that the higher-order moments are neglected. Thus we can discard them from the calculation at this point. However, it should be noted that there is a great deal of work that investigates the effect of preserving the higher-order moments in the calculation. One might imagine that the more information that is retained, the better the approximation should work.

To simplify the notation, we define the following two quantities

$$P_{k+1} = E[(x_{k+1} - x_{0_{k+1}})^2] \tag{21.2.7a}$$

$$P_k = E[(x_k - x_{0_k})^2] \tag{21.2.7b}$$

which are measures of the error variance between the true solution and the prediction at t_{k+1}, and the true solution and our best estimate of it at t_k respectively. This notation gives

$$P_{k+1} = P_k(f'(x_{0_k}))^2 + E[q_{k+1}^2] \tag{21.2.8}$$

Note that P_{k+1} accounts for the dynamics (and errors associated with it) while P_k accounts for errors in estimating the initial state.

To make a data assimilated prediction, $\bar{x}_{k+1}$, of the state of the system at time t_{k+1} using an observation y_{k+1}, we then apply the following formula

$$\bar{x}_{k+1} = x_{0_{k+1}} + K_{k+1}(y_{k+1} - x_{0_{k+1}}) \tag{21.2.9}$$

where the innovation is again given by the quantity $y_{k+1} - x_{0_{k+1}}$ and the Kalman gain K_{k+1} is given by

$$K_{k+1} = \frac{P_{k+1}}{P_{k+1} + R} \tag{21.2.10}$$

where R is the observation error variance. If there is no observational error, then $R = 0$ which forces $K_{k+1} = 1$. This in turn gives the assimilated prediction to be $\bar{x}_{k+1} = y_{k+1}$, i.e. the assimilated prediction is exactly the error free experimental observation. The variance associated with the error of our prediction $\bar{x}_{k+1}$ is given by

$$\bar{P}_{k+1} = (1 - K_{k+1})P_{k+1} \tag{21.2.11}$$

Taken together, the data assimilation results, which now directly tie in the dynamics of the system through (21.2.2), define what is called the *extended Kalman filter* (EKF).

The Vector EKF

The vector version of the above one-dimensional argument follows in a straightforward manner [84, 87]. The dynamics given by (21.2.2) and (21.2.3) are now

written

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k) + \mathbf{q}_{k+1} \tag{21.2.12}$$

and

$$\mathbf{x}_{0_{k+1}} = f(\mathbf{x}_{0_k}) \tag{21.2.13}$$

respectively. Once again Taylor expanding now yields the covariance evolution

$$\mathbf{P}_{k+1} = \mathbf{J}(\mathbf{f})\mathbf{P}_k\mathbf{J}(\mathbf{f})^T + \mathbf{Q} \tag{21.2.14}$$

where $\mathbf{J}(\mathbf{f})$ is the Jacobian generated form the multi-dimensional Taylor expansion and higher-order moments have been neglected. This formula is equivalent to (21.2.8).

Given a data observation $\mathbf{y}_{k+1}$ at time t_{k+1}, the vector case requires an appropriate mapping of the observation space to the state space. But this was already discussed and is mathematically transcribed in (21.2.1). Finally, the data assimilated prediction of the correct state is given by

$$\bar{\mathbf{x}}_{k+1} = \mathbf{x}_{0_{k+1}} + \mathbf{K}_{k+1}(\mathbf{y}_{k+1} - \mathbf{H}\mathbf{x}_{0_{k+1}}) \tag{21.2.15}$$

where $\mathbf{K}_{k+1}$ is the Kalman gain matrix given by

$$\mathbf{K}_{k+1} = \mathbf{P}_{k+1}\mathbf{H}^T(\mathbf{H}\mathbf{P}_{k+1}\mathbf{H}^T + \mathbf{R})^{-1} \tag{21.2.16}$$

and $\mathbf{R}$ is the noise covariance matrix. The error covariance of the updated state vector is given by

$$\bar{\mathbf{P}}_{k+1} = (\mathbf{I} - \mathbf{K}_{k+1}\mathbf{H})\mathbf{P}_{k+1} \tag{21.2.17}$$

where $\mathbf{I}$ is the identity matrix. In vector form, the innovation is given by $\mathbf{y}_{k+1} - \mathbf{H}\mathbf{x}_{0_{k+1}}$. Thus the matrix $\mathbf{H}$ serves an important role in overlaying the data measurement locations with the underlying grid used for computationally evolving forward the model dynamics. This is the EKF for generic complex systems.

With the EKF now established, its strengths and weaknesses are briefly considered. Its strengths are obvious, the EKF provides a systematic way to integrate observational data into the modeling process, thus improving the model predictions. Its most obvious drawbacks are computational. Specifically, for complex systems where the state vector is defined by potentially millions or billions of variables on a large computational grid, computing the innovation becomes unwieldy, especially as these large matrices need to be inverted and Jacobians found. Such enormous computational expense can render the method useless from a practical point of view. One potential way to deal with such computational complexity is to minimize (21.1.6) directly using, for instance, gradient descent algorithms. This is ultimately faster than computing Jacobians and inverses of the large matrices under consideration. Another potential way to solve the problem is to use Ensemble Kalman Filtering (EnKF) techniques which render the problem tractable by processing observations one at a

time or by breaking the domain into smaller subdomains where the matrices are tractable. Such computational considerations are necessary to consider in the types of systems (weather prediction and climate modeling [82]) where data assimilation has becomes a standard method of analysis.

21.3 Data Assimilation for the Lorenz Equation

To demonstrate the data assimilation algorithm in practice, consideration will be given to the Lorenz equations

$$x' = \sigma(y - x) \tag{21.3.1a}$$
$$y' = rx - y - xz \tag{21.3.1b}$$
$$z' = xy - bz \tag{21.3.1c}$$

Thus the vector $\mathbf{x} = [x\ y\ z]^T$ is the three-degree of freedom state vector whose nonlinear evolution $f(\mathbf{x})$ is specified by the right hand side (Lorenz model) in (21.3.1) . The Lorenz equations are a highly simplified model of convective-driven atmospheric motion (See Sec. 24.3 for a derivation of the system).

To begin, a perfect simulation will be performed of the Lorenz system, i.e. a simulation that includes specified initial conditions without noise and an evolution which is free of stochastic forcing. This simulation will be the *truth* that the data assimilation will try to reproduce. The parameters to be simulated here, and in what follows, are standard in many examples: $\sigma = 10$, $b = 8/3$ and $r = 28$. The large value of r puts the system in a dynamical state that is highly sensitive to initial conditions. The perfect initial conditions will be $\mathbf{x}_0 = [5\ \ 5\ 5]^T$. The following MATLAB code solves this problem for the time domain $t \in [0, 20]$ and plots it in a parametric way in 3D.

```
t=0:0.01:20;
sigma=10; b=8/3; r=28;

x0=[5 5 5];
[t,xsol]=ode45('lor_rhs',t,x0,[],sigma,b,r)

x_true=xsol(:,1); y_true=xsol(:,2); z_true=xsol(:,3);
figure(1), plot3(x_true,y_true,z_true)
```

The right-hand side of the differential equation includes the dynamics through $f(\mathbf{x})$. In this case, the function **lor_rhs** is given by

```
function rhs=lor_rhs(t,x,dummy,sigma,b,r)
rhs=[sigma*(-x(1)+x(2))
    -x(1)*x(3)+r*x(1)-x(2)
    x(1)*x(2)-b*x(3)];
```

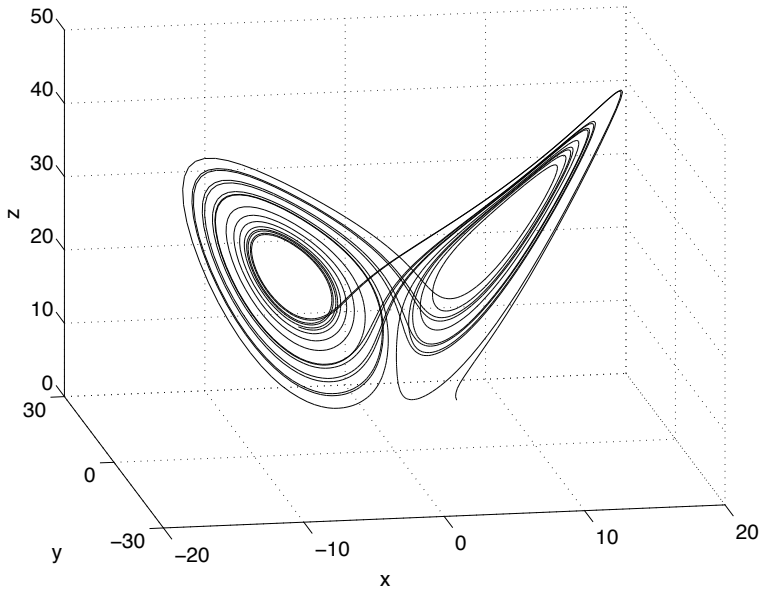

Figure 239: Evolution of the variables $x(t)$, $y(t)$ and $z(t)$ over the time period $t \in [0, 20]$ for $\sigma = 10$, $b = 8/3$ and $r = 28$. The evolution is shown parametrically as a function of time.

The results of simulating the Lorenz equation are shown in Figs. 239 and 240. The first of these figures demonstrates the standard *butterfly* pattern of evolution of the strange attractor in three dimensions. In this graph, time is a parametric quantity. The second of these figures illustrates the evolution of the variables $x(t)$, $y(t)$ and $z(t)$ over the time period $t \in [0, 20]$. In what follows, instead of tracking and comparing all the variables, we will focus on $x(t)$ for illustrative purposes only.

Sensitivity to initial conditions

The first thing that will be investigated is sensitivity of the evolution to small changes in the initial conditions. The mathematical statement of this problem is given in (21.1.2). Thus the effect of $\mathbf{q}_2$ on the dynamics will be considered. And in particular, it is the perturbation of the initial conditions that compromises the predictive power of the theoretical model. To simplify this, we will assume that the error has a Gaussian distribution so that

$$\mathbf{x}(0) = \mathbf{x}_0 + \sigma_2 \mathbf{q}(0, 1) \tag{21.3.2}$$

where $\mathbf{x}_0$ is the perfect initial conditions and $\mathbf{q}(0, 1)$ is a Gaussian distributed random variable with mean zero and unit variance. Thus σ_2 is chosen to make

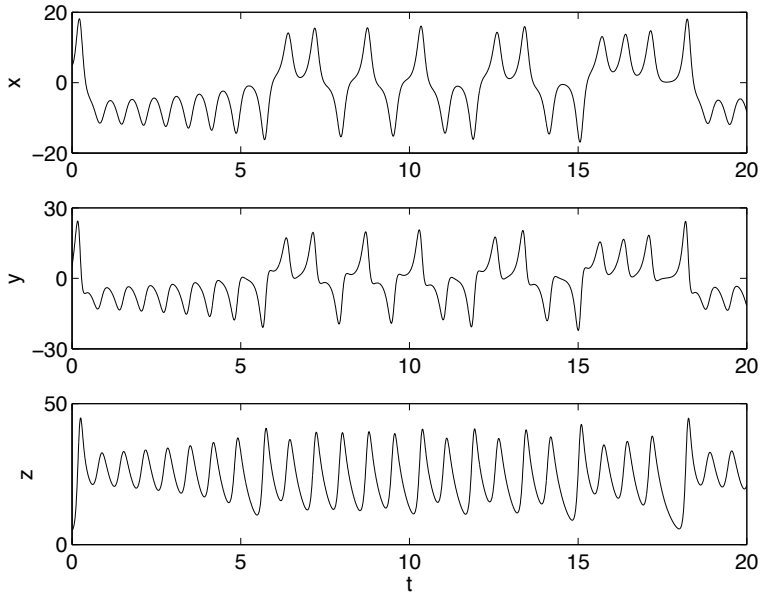

Figure 240: Time evolution of the variables $x(t)$, $y(t)$ and $z(t)$ over the time period $t \in [0, 20]$ for $\sigma = 10$, $b = 8/3$, $r = 28$ and $\mathbf{x}_0 = [5\ 5\ 5]^T$. In this parameter regime, specifically for this large a value of r, the dynamics of the Lorenz equations are highly sensitive to initial conditions.

the error variance either larger or smaller.

With this error, the evolution of (21.3.1) can once again be explored. In Fig. 241, eight realizations of the evolution are given using the initial conditions (21.3.2). The exact evolution which we are trying to model (with $\sigma_2 = 0$) is the thinner solid line while the initial conditions with error (with $\sigma_2 = 1$) is the bolded line. The following MATLAB code generates the eight realizations:

```
sigma2=1;  % error variance
for j=1:8
  xic=x0+sigma2*randn(1,3); % perturb initial conditions
  [t,xsol]=ode45('lor_rhs',t,xic,[],sigma,b,r);
  x=xsol(:,1);  % projected x values
  subplot(4,2,j), plot(t,x_true,'k'), hold on
  plot(t,x,'k','Linewidth',[2])
end
```

For all these realizations, the projected state fails to model the *true* dynamics after $t \approx 5$. Indeed, after this time, there is almost no correlation of closeness

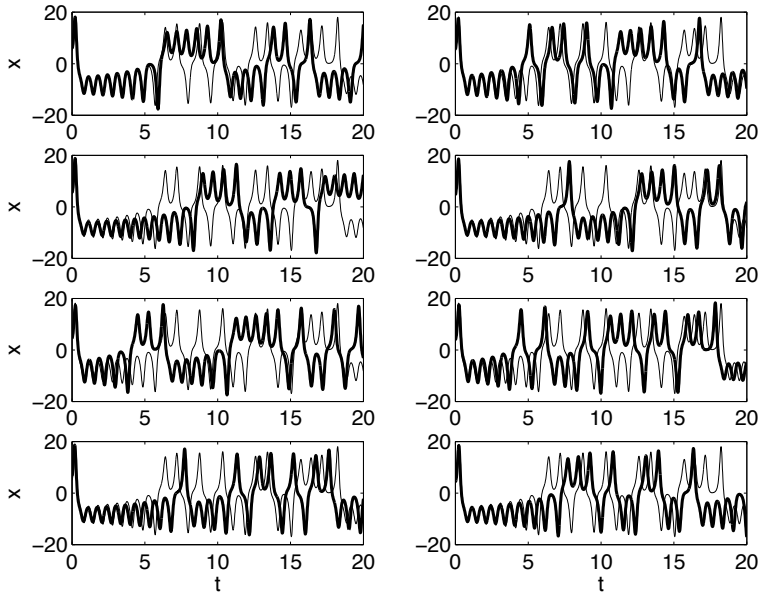

Figure 241: Time evolution of the variables $x(t)$, $y(t)$ and $z(t)$ over the time period $t \in [0, 20]$ for $\sigma = 10$, $b = 8/3$, $r = 28$ and $\mathbf{x}_0 = [5\ 5\ 5]^T$. Here, eight realizations are shown for the perturbed initial conditions given by (21.3.2) with $\sigma_2 = 1$. Note that for all simulations, after $t \approx 5$ the true dynamics (light line) differ from the dynamics with perturbed initial conditions (bold line). Thus prediction of the dynamics beyond this time is virtually impossible given such perturbations (errors) in the initial data.

between the truth and our projection based upon noisy initial data. This illustrates the need for data assimilation. Specifically, it is hoped that occasional measurements of the data would allow for an accurate prediction of the true future state far beyond $t \approx 5$.

Data assimilation for the Lorenz equations

The goal in this example will be to simply illustrate the EKF algorithm. Thus a simple case will be taken which can be completely coded in MATLAB in a fairly straightforward manner. In this simple example, no stochastic forcing of the differential equations will be considered so that the dynamics are *exactly* as specified in (21.3.1). Said another way, the error vector $\mathbf{q}_1$ in (21.1.6) is zero. However, there will be both error in the measurements ($\mathbf{q}_3$) and the initial conditions ($\mathbf{q}_2$). Knowing full well about the sensitivity to initial conditions in the Lorenz equations, the initial noise will greatly compromise the ability

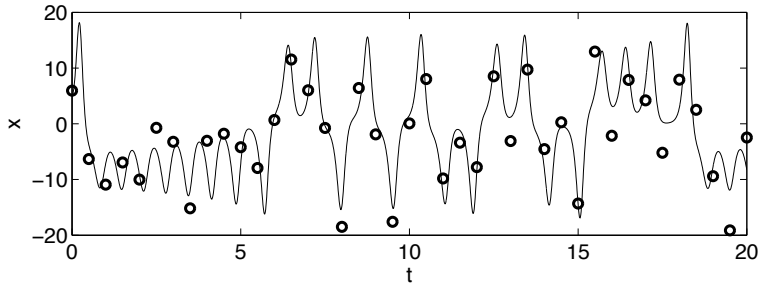

Figure 242: True time dynamics of the variable $x(t)$ over the time period $t \in [0, 20]$ for $\sigma = 10$, $b = 8/3$, $r = 28$ and $\mathbf{x}_0 = [5\ 5\ 5]^T$. The circles represent experimental measurements at every half-unit of time of the true dynamics with error given by (21.3.3) with $\sigma_3 = 4$. The dynamics of the variables $y(t)$ and $z(t)$ are similar. Data assimilation makes use of the experimental measurements to keep the model predictions closer to the true dynamics.

of the model to predict the future state of the system. The hope is that data assimilation will mitigate this problem to some extent and allow for much more accurate, and longer time, predictions for the future state of the system.

In our example, the data collection points will be at simulation time points already specified by our differential equation solver. Moreover, data will be taken from all variables. Thus the mapping matrix $\mathbf{H} = \mathbf{I}$. To illustrate the data collection process, consider then modification of (21.2.1) to

$$\mathbf{y}(t_n) = \mathbf{x}(t_n) + \sigma_3 \mathbf{q}(0, 1) \tag{21.3.3}$$

where t_n is a measurement time point and $\mathbf{q}(0, 1)$ is a Gaussian distributed random variable with mean zero and unit variance. Thus σ_3 is chosen to make the error variance either larger or smaller in the data measurements. Figure 242 shows the true dynamics (line) and data collected every half-unit of time with $\sigma_3 = 4$ (circles). As is clearly seen, the data is collected under error and does not match up perfectly with the true dynamics. However, it does follow the true dynamics fairly closely over the time period of integration. To compute these data points in MATLAB, the following code is used in conjunction with the code that generates the true dynamics:

```
% noisy obserations every t=0.5
tdata=t(1:50:end);
n=length(tdata)
xn=randn(n,1); yn=randn(n,1); zn=randn(n,1);
sigma3=4;  % error variance in data
xdata=x(1:50:end)+sigma3*xn;
ydata=y(1:50:end)+sigma3*yn;
```

```
zdata=z(1:50:end)+sigma3*zn;
```

The idea is to use these experimental points along with the noisy initial conditions shown in Fig. 241 in order to enhance our prediction for the future state.

Given that $\mathbf{H} = \mathbf{I}$ in this example, this reduces the EKF algorithm to computing

$$\bar{\mathbf{x}}_{k+1} = \mathbf{x}_{0_{k+1}} + \mathbf{K}_{k+1}(\mathbf{y}_{k+1} - \mathbf{x}_{0_{k+1}}) \tag{21.3.4}$$

where $\mathbf{K}_{k+1}$ is the Kalman gain matrix given by

$$\mathbf{K}_{k+1} = \mathbf{P}_{k+1}(\mathbf{P}_{k+1} + \mathbf{R})^{-1} \tag{21.3.5}$$

and $\mathbf{R}$ is the noise covariance matrix. The error covariance of the updated state vector is given by

$$\bar{\mathbf{P}}_{k+1} = (\mathbf{I} - \mathbf{K}_{k+1})\mathbf{P}_{k+1}\,. \tag{21.3.6}$$

Note that the matrices involved are 3×3 matrices and the innovation is simply given by $\mathbf{y}_{k+1} - \mathbf{x}_{0_{k+1}}$. From (21.2.14), and using the fact that the dynamics propagates in an error free fashion ($\mathbf{q}_1 = 0$), then

$$\mathbf{P}_{k+1} = \mathbf{J}(\mathbf{f})\mathbf{P}_k\mathbf{J}(\mathbf{f})^T\,. \tag{21.3.7}$$

where the Jacobian for the Lorenz equation can be easily computed to give

$$\mathbf{J}(\mathbf{f}) = \begin{bmatrix} -\sigma & \sigma & 0 \\ r - z & -1 & -x \\ y & x & -b \end{bmatrix} \tag{21.3.8}$$

and the matrix $\mathbf{P}_k$ measures the error in estimating the initial state of the system at time t_k. This error is determined by (21.3.2) and the parameter σ_2.

Everything is in place then to implement the data assimilation procedure. The following is an algorithmic outline of what needs to occurs:

(i) Determine the sources of error and how to incorporate them. The error in the data measurement determines the matrix $\mathbf{R}$ while the error in the initial condition determines the matrix $\mathbf{P}_k$ (Note that we are ignoring errors generated in the dynamics themselves).

(ii) Compute the Jacobian at time t_k using the best estimate for the state vector $\mathbf{x}_0(t_k)$ and combine it with the computation of $\mathbf{P}_k$ in order to compute $\mathbf{P}_{k+1}$.

(iii) With $\mathbf{P}_{k+1}$ and $\mathbf{R}$, compute the Kalman gain matrix $\mathbf{K}_{k+1}$.

(iv) Compute the new state of the system using the innovation vector and the Kalman gain matrix.

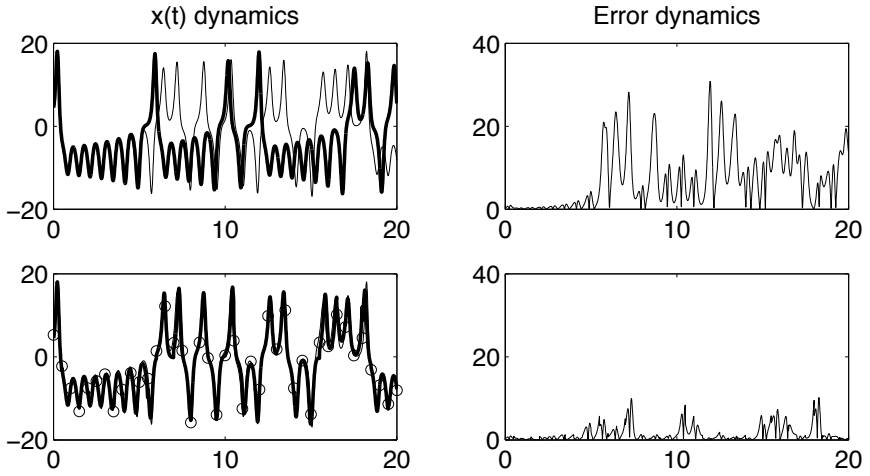

Figure 243: Comparison of the model dynamics using a direct numerical simulation of the noisy initial conditions (top panels) with the data assimilated solution with noisy initial conditions and noisy data measurements (bottom panels). The direct simulation (bold line) fails to predict the true dynamics (line) beyond $t \approx 5$ (top panel). Indeed, the error in this case grows quite large at this time (top right panel). When making use of the data assimilation technique (bottom panels) and the data measurements (circles), the solution stays close to the true solution for a much longer time with much smaller error (bottom right panel). Thus data assimilation can greatly extend the time window under which the model can be useful.

(v) Use the new state of the system to again project to another time into the future where observational data is once again available.

The Lorenz equation is a fairly trivial example to consider. Moreover, our treatment of the data assimilation will be for the simplest case possible. In this example, the error variance for both the initial conditions and data measurements will both be unity so that $\sigma_2 = \sigma_3 = 1$ respectively. The Kalman gain matrix will then be given as in the one-dimensional case: $K = \sigma_2/(\sigma_2 + \sigma_3)$. A code for simulating this system and making adjustments based upon the data measurements is as follows:

```
x_da=[];  % data assimilation solution
for j=1:length(tdata)-1  %  step through every t=0.5
  tspan=0:0.01:0.5;  % time between data collection
  [tspan,xsol]=ode45('lor_rhs',tspan,xic,[],sigma,b,r);

  xic0=[xsol(end,1); xsol(end,2); xsol(end,3)]  % model estimate
```

```
  xdat=[xdata(j+1); ydata(j+1); zdata(j+1)] % data estimate
  K=sigma2/(sigma2+sigma3);  % Kalman gain
  xic=xic0+(K*[xdat-xic0])  %  adjusted state vector

  x_da=[x_da; xsol(1:end-1,:)];  % store the data
end
x_da=[x_da; xsol(end,:)];  % store last data time
```

In this simulation, the vector **data-xic0** is the innovation vector that is weighted according to the Kalman gain matrix. Figure 243 shows the results of this simulation for one representative realization of the error vectors. In the top panels, the non-data assimilated computation is shown showing that the simulation solution diverges from the true solution around $t \approx 5$ (See also Fig. 241). The error between the true solution and the model solution is shown in the right panel. Note that the error is quite large around $t \approx 5$, thus making any prediction beyond this time fairly useless. In the bottom panel, the data assimilated solution is demonstrated (bold line) and compared to the true dynamics (line). The data assimilated solution is nearly indistinguishable from the true dynamics. The experimental observations are shown by circles at every half-unit of time. The error between the data assimilated solution and true dynamics is shown in the right panel. Note that the error remains quite small in comparison to the direct simulation from noisy initial data. Regardless, there is a build up of error that will eventually grow large as the data assimilated solution also diverges from the true dynamics. Ultimately, the data assimilated solution allows for significant extension of the time window where the model prediction is valid.

Data assimilation, in general, is much more sophisticated than what has been applied here to a simple 3×3 system. Indeed, for highly complex systems, the model error in the dynamics plays a fundamental role characterizing the behavior in addition to measurement and initial condition error. How one chooses to not only treat this error, but how to sample from the dynamics in time, gives rise to a great variety of mathematical techniques for enhancing the data assimilation method [83, 84, 85, 86]. Such data assimilation methods are at the heart of cutting-edge technology, for instance, in weather prediction and/or climate modeling. The hope here was simply to illustrate the basic ideas of this tremendously powerful methodology.

22 Equation Free Modeling

Multiscale phenomena, and their associated complex systems, are an increasingly important class of problems that need to be addressed from a mathematical point of view. As illustrated previously in the lecture on wavelets, some mathematical progress has been made in separating differing length scales by more sophisticated (wavelet) transforms. Alternatively, one can think about